GC20-1734-2

Systems

A Guide to the IBM System/370 Model 145

This guide presents hardware, programming systems, and other pertinent information about the IBM System/370 Model 145 that describes its significant new features and advantages. The contents are intended to acquaint the reader with the Model 145 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 145 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**
- *OS/Virtual Storage 1 Features Supplement* (GC20-1752)
- *OS/Virtual Storage 2 Features Supplement* (GC20-1753)
- *Virtual Machine Facility/370 Features Supplement**

*Availability to be announced

IBM

Third Edition (August 1972)

This is a major revision obsoleting GC20-1734-1. Text has been added or changed to include information about the following: EC mode; dynamic address translation; channel indirect data addressing; program event recording; CPU timer and clock comparator; new System/370 instructions; store status facility; integrated storage control for 3330-series disk storage; 3830 Storage Control Model 2; virtual storage concepts, advantages, terminology, and performance factors; DOS/VS, OS/VS1, and OS/VS2 support of Model 145 hardware features; DOS Version 4 support; and miscellaneous additions and corrections. Changes to the text and illustrations are denoted by the vertical line to the left of the change. In addition, this publication is now structured as a base publication for which optional programming systems supplements describing virtual storage operating systems can be ordered.

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 145, 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 Department, 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/360. The reader should have a general knowledge of System/360 architecture, channels, I/O devices, and programming systems support. This guide highlights only those Model 145 hardware, I/O, and programming systems features that are different from those of System/360 models and discusses their significance. This publication applies to systems with 60-cycle power.

The total Model 145 guide consists of this base publication (Sections 01 to 70), which covers virtual storage concepts and Model 145 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 operating systems that support a virtual storage environment using the dynamic address translation hardware of the Model 145. 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* - assumes knowledge of DOS Version 4

- OS/Virtual Storage 1 Features Supplement (GC20-1752) - assumes knowledge of OS MFT

- OS/Virtual Storage 2 Features Supplement (GC20-1753) - assumes knowledge of OS MVT

- Virtual Machine Facility/370 Features Supplement*

All optional supplements also assume knowledge of virtual storage, dynamic address translation, and other new Model 145 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.

*Availability to be announced

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 (*)	OS/VS1 Features Supplement (GC20-1752)	OS/VS2 Features Supplement (CGC20-1753)	VM/370 Features Supplement (*)
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

Additional, more detailed information regarding System/370 Model 145 hardware and programming systems support can be found in system library publications.

*Availability to be announced

CONTENTS

Base Publication Sections (01 to 70)

<u>Optional</u> <u>Sections</u> (See each supplement for detailed contents and index)

<u>FIGURES</u> (Sections 01 to 70)

TABLES (Sections 01 to 70)

SECTION 01: SYSTEM HIGHLIGHTS

The System/370 Model 145 is designed to enhance, extend, and broaden the successful concepts of System/360 and to provide significant new functions that do not necessitate a major reprogramming effort. It is a general purpose growth system for System/360 Model 40 and large Model 30 users that offers significant price performance improvement in addition to many new features. The System/370 Model 145 retains and extends the wide range of commercial and scientific data processing capabilities offered by System/360 Models 30 and 40. It also offers functions designed to facilitate new application development and to ease entry into, and expansion of, online data processing operations. The Model 145 is compatible with other System/370 models.

Transition from System/360 models to the System/370 Model 145 can be accomplished with a minimum of effort because most current System/360 user programs, I/O devices, and programming systems are upward compatible with the new system. Additional capabilities are included in OS MFT and MVT and in DOS Versions 3 and 4 to support certain new features of the Model 145, thereby providing proven operating system performance as well as continuity.

Compatible growth from System/360 operating systems to a Model 145 virtual storage environment can be achieved using the new 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 new operating systems will run only on System/370 models with extended System/370 functions, namely on those with extended control mode of system operation and dynamic address translation facilities. In addition to supporting virtual storage, the System/370 operating systems offer several 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 for System/370. While CP/67 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.

Transition with little or no reprogramming is also provided for 1401/1440/1460 and 1410/7010 users and for those presently emulating these systems on System/360. Improved emulators for these systems that operate under OS (MFT, MVT, VS1, and VS2) or DOS (Versions 3, 4, and VS) control on the Model 145 are available.

DOS users who wish to install OS on their Model 145 can ease the transition by using the standard OS/DOS Compatibility feature. An OS DOS Emulator program is provided that supports emulation of a DOS Version 3 or 4 multiprogramming system under OS (MFT, MVT, VS1, or VS2) control.

Highlights of the Model 145 are as follows:

- Upward compatibility with most System/360 architecture and programming has been maintained through implementation of the basic control (BC) mode of system operation. An extended control (EC) mode of operation, not implemented in System/360, is also provided.

- Internal performance of a Model 145 operating in BC mode is from approximately three to five times that of the Model 40 for a typical instruction mix.

- The following are CPU features of the Model 145.

Implementation of a basic control mode and an extended control mode of system operation is standard. When the Model 145 is operating in BC mode, DOS Versions 3 and 4, OS MFT, and OS MVT can be used. When the Model 145 operates in EC mode, the PSW format and the layout of permanently assigned lower processor storage are altered to support additional system control and new functions (such as dynamic address translation) that are not provided in BC mode. DOS/VS, OS/VS1, OS/VS2, and VM/370 support EC mode operations. With a few exceptions, existing user-written problem programs that operate under DOS Version 4, OS MFT, or OS MVT can operate under DOS/VS, OS/VS1, or OS/VS2, respectively, without modification.

Dynamic address translation (DAT) is a standard facility that can operate only in EC mode. It provides 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 size of the virtual storage that can be efficiently supported on a Model 145 depends on the hardware configuration and job stream characteristics.) DAT is a standard feature on all System/370 models that have EC mode. In System/360, DAT is implemented only in the Model 67.

Channel indirect data addressing is also standard and is provided to handle I/O operations when DAT is used, because an I/O buffer area can span a set of noncontiguous processor storage areas.

The Model 145 standard instruction set includes new general purpose instructions in addition to the powerful System/360 instruction set. These instructions enhance decimal arithmetic performance, simplify the handling of nonword-size data that is processed using the general registers, eliminate the need for multiple move or compare instructions or move subroutines, simplify system mask handling, and facilitate record blocking and deblocking, field padding, and storage clearing. Other new instructions are provided primarily for control program use.

A floating-point arithmetic option is available that provides for floating-point operations, including extended precision operations. Precision of up to 28 hexadecimal digits, equal to up to 34 decimal digits, is provided by the extended precision data format.

A monitoring feature is standard that can be used to trace user-defined program events for the purpose of debugging or statistics gathering.

Program event recording is standard and also is designed to be used as a problem determination aid. This feature includes hardware that monitors the following during instruction execution: successful branches, the alteration of general registers, and instruction fetching from and alterations of specified areas of processor storage. It can operate only when the Model 145 is in EC mode.

An interval timer of 3.33-ms resolution to improve job accounting accuracy is standard. (A 16.6-ms resolution timer is provided for Models 30 and 40.)

A time of day clock is included as a standard feature to provide more accurate time of day values than does the interval timer. The clock has a 1-microsecond resolution.

The CPU timer and the clock comparator are optional. 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 standard interval timer and enable more efficient timing facility routines to be written.

The byte-oriented operands facility permits byte boundary alignment for the operands of nonprivileged instructions to eliminate the necessity of adding padding bytes within records or to blocked records for the purpose of aligning fixed- or floating-point data.

Retry of failing CPU operations is handled automatically by the hardware, without programming assistance.

- Functionally improved relocatable emulators are available that operate under operating system control. Concurrent execution of System/370 programs with any combination of 1401, 1440, 1460, 1410, and 7010 programs in a multiprogramming environment is supported. The 1401/1440/1460 and the 1401/40/60, 1410/7010 Compatibility features are optional, no-charge features. These emulators are supported by DOS Versions 3 and 4, DOS/VS, and OS MFT, MVT, VS1, and VS2.

- The standard OS/DOS Compatibility feature permits emulation of a DOS Version 3 or 4 system under OS (MFT, MVT, VS1, or VS2), concurrently with execution of other OS jobs. Both DOS emulation and 1400/7010 emulation can operate together on a Model 145 under OS control.

- Operator console devices with an alter/display mode are available.

  The 15-cps 3210 Model 1 Console Printer-Keyboard or the 85-cps 3215 Model 1 Console Printer-Keyboard is required. A remote 3210 Model 2 Console Printer-Keyboard can be installed in addition to either of the other printer-keyboards.

- The following channel features are available for the Model 145:

  Up to four high-speed selector channels can be attached (one selector is standard) in addition to the standard byte multiplexer channel. Up to 256 subchannels can be installed on the byte multiplexer channel. An individual selector channel can operate at a data rate of .82 megabytes per second (MB). With a special feature, a selector channel can operate at a 1.85-MB data rate and therefore support significantly faster I/O devices than can Model 30 and 40 channels.

  Block multiplexer mode of operation for selector channels is a no-charge optional feature. A block multiplexer channel is a superset of a selector channel. When used in conjunction with rotational position sensing devices, such as 3330-series and 2305 Model 2 direct access storage devices, block multiplexer channels can increase total system throughput by permitting increased amounts of data to enter and leave the system in a given time period. A single block multiplexer channel can support interleaved, concurrent execution of multiple high-speed I/O operations.

  The optional Integrated File Adapter feature allows lower cost, direct attachment of 2314A-type disk drives to a Model 145. A selector channel and a disk control unit are not required. The 2319 Disk Storage units (Models A1 and A2 with three drives each), 2312 Disk Storage (one drive), 2313 Disk Storage (four drives), and 2318 Disk Storage (two drives) can be connected to the adapter for a configuration of from three to eight natively attached disk drives. Facilities consisting of from one to eight 2314-type disk drives, plus a spare, can also be channel attached to the Model 145.

Channel retry data is provided when channel errors occur so that error recovery routines can retry I/O operations.

- The following significant new storage features are provided by the Model 145:

  All system storage--local, control, and processor (main)--is implemented using monolithic technology instead of discrete ferrite cores.

  > From 160K to 512K of monolithic processor (main) storage is available--twice the maximum main storage available on the Model 40. Four data bytes can be fetched in 540 nanoseconds, while four data bytes can be stored in 607.5 nanoseconds.
  >
  > Reloadable, monolithic control storage is used to contain the microcode necessary for system operation. Control storage contains the microcode required for standard and optional features and can be expanded from 32K to 64K in 2K increments, as required. Use of writable, instead of read-only, control storage offers the advantages of system cost reduction and improved system serviceability.

  Error checking and correction (ECC) hardware, which automatically corrects all single-bit processor and control storage errors and detects all double-bit and most multiple-bit errors, is standard.

- I/O devices for the Model 145 include the following.

  Most I/O devices for System/360 Models 25, 30, and 40 can be attached.

  The 3505 Card Reader and the 3525 Card Punch with optional card read capability can be attached. A variety of models are available. They offer 80-column card users configuration flexibility, new functions, high reliability, and greatly expanded error recovery facilities.

  > Models B1 and B2 of the 3505 Card Reader can operate at 800 and 1200 cards per minute, respectively. Significant new features include Optical Mark Reading (optional) and Read Column Eliminate (standard). The latter permits successful reading of cards containing internal perforations or other holes that would normally cause an error.
  >
  > Models P1, P2, and P3 of the 3525 Card Punch can punch and, optionally, read 100, 200, and 300 cards per minute, respectively. New features of this unit include automatic punch retry when an error is detected (standard) and card printing. A two-line print option and a multiline (up to 25 lines) print option are available.

  The 3803/3420 Magnetic Tape Subsystem is attachable. Models 3, 5, and 7 of the 3420 Magnetic Tape Unit, with data rates of 120 KB, 200 KB, and 320 KB, respectively, at 1600-BPI recording density, are provided. Phase-encoded recording, which permits automatic correction of all single-bit read errors in flight, is used. This new tape subsystem offers improved price performance; Dual Density and Seven Track features for compatibility with, and conversion of, 2400-series tape volumes; greatly reduced operator handling through implementation of such features as automatic tape threading and cartridge loading; lower cost tape switching than is provided for 2400-series tape units; and enhanced reliability, availability, and serviceability features.

The 3410/3411 Magnetic Tape Subsystem, Models 1, 2, and 3, can be attached to provide data rates of 20 KB, 40 KB, and 80 KB, respectively, at 1600-BPI density. Phase-encoded recording is used. A Model 1 subsystem can consist of from one to four tape units. Models 2 and 3 of the subsystem can have from one to six tape units. This subsystem offers improved price performance for data rates under 120 KB, a simplified tape path to speed tape setup, Dual Density and Seven Track features, a totally new compact physical design that minimizes floor space requirements, and reliability, availability, and serviceability improvements.

The high-speed 3211 Printer, with a tapeless carriage and print speed of 2000 alphameric lines per minute, is attachable. The tapeless carriage decreases operator intervention by eliminating carriage tape loading and unloading.

The 3330-series disk storage can be attached via 3830 Storage Control (Models 1 and 2) and via Integrated Storage Control. The 3330-series offers significantly faster seeks and more than twice the data rate of the 2314. Eight 3330-series drives offer more than three times the capacity of eight 2314 drives. Automatic error correction features, rotational position sensing, and multiple requesting are also provided as standard features.

The 3330 has an 806-KB data transfer rate, average seek time of 30 ms, and full rotation time of 16.7 ms. A 3330-series drive has a maximum capacity of 100 million bytes. A string of two, four, six, or eight 3330-series drives can be configured. Model 1 of 3830 Storage Control can handle one string of up to eight drives. Model 2 of 3830 Storage Control and Integrated Storage Control can handle one or two strings of up to eight drives each.

The 2305 facility Model 2, with a maximum single module capacity of 11.2 million bytes, a data rate of 1.5 MB, and an average access of 5 ms, can be attached to a Model 145 to be used as a system residence device or as high-speed storage. One 2305 Model 2 facility can include two modules for a maximum facility capacity of 22.4 million bytes.

- Extensive hardware and programming systems error recovery and repair features are provided to enhance system reliability, availability, and serviceability.
- Compact physical design reduces Model 145 space requirements. The Model 145 has almost three times the number of circuits as a Model 40, yet a 256K, five-channel system requires about the same amount of space as a 256K, three-channel Model 40.

As the highlights indicate, Model 30 and 40 users now have a broader range of Model 145 configurations from which to choose when tailoring a growth system with improved throughput and expanded capabilities. Specifically, the Model 145 offers the following advantages when compared to Models 30 and 40.

Larger, Faster Processor (Main) Storage Sizes

Processor storage sizes of 160K, 208K, 256K, 384K, and 512K bytes are provided. The Model 30 can have a maximum of 64K, while 256K is the largest main storage size available for a Model 40. The cycle time of Model 145 processor storage is about four times faster than that of the Model 40. This improved cycle time increases internal performance and permits faster I/O devices to be attached to the system.

Additional storage can contribute significantly to system capabilities and performance. Specifically, the addition of more processor storage provides the Model 145 user with the ability to:

- Execute more or larger jobs concurrently, including new application and integrated emulator jobs

- Add and expand applications, such as graphics, teleprocessing, time sharing, and data base, that require larger amounts of storage

- Use higher level language translators and linkage editors that provide more functions and execute faster

- Execute larger processing programs without the necessity of overlay structures

- Allocate more storage to language translators and sorts to improve their execution speed

- Use more and larger I/O buffers to speed up input/output operations and optimize use of direct access storage and tape media space

- Include system generation options that improve control program performance and support additional functions

Support of a Virtual Storage Environment

While the Model 145 has larger processor storage than its comparable-scale System/360 models, it also provides the capability of implementing 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, many of the 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 many of the new hardware features and I/O devices for the Model 145 and 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.

Greatly Expanded Channel Capabilities

The fast internal performance of the Model 145, together with expanded use of multiprogramming, requires that more data be available faster than on the Model 40. A variety of channel options are provided.

Twice the number of byte multiplexer subchannels can be installed on a Model 145 as on a Model 40. The Model 145 also offers more and faster high-speed channels than Models 30 and 40, and block multiplexer channels not provided for these System/360 models. The basic individual channel data rate of .82 MB can be more than doubled by installation of the optional Channel Word Buffer feature.

The channel features of the Model 145 provide:

- Up to 256 byte multiplexer subchannels for larger teleprocessing users

- Lower cost direct attachment of 2314A-type disk storage drives via the new Integrated File Adapter

- Attachment of high-speed direct access devices such as the 3330-series and 2305 Model 2

- Potential increases in channel throughput via use of block multiplexing and rotational position sensing to improve effective data transfer rates

- A significantly higher attainable aggregate channel data rate than the Model 40 to balance the higher performance capabilities of the Model 145 CPU

<u>Faster I/O Devices and Increased Direct Access Storage Capacity</u>

The Model 145 supports a faster magnetic tape unit than do Models 30 and 40--specifically, the 3420 Model 7 with a data rate of 320 KB.

A Model 145 I/O configuration can also include significantly more and faster direct access storage. For example, the Model 145 is not limited to having 2314 facilities on only one channel, as are Models 30 and 40. In addition, the 3330-series and 2305 Model 2 provide considerably more capacity and faster data access than 2314 facilities or 2303 Drum Storage because of higher data transfer rates, faster rotation, and new features. These direct access devices also offer higher availability through use of new hardware-only and program-assisted error correction features.

The 3330-series provides large capacity and fast access for a lower cost per bit. It is à growth device that offers improved price performance for the 2314 facility and the 2321 Data Cell Drive. The 3330-series is designed to be used in every area in which direct access storage is needed, for example:

- As a system residence device and for program library storage

- In teleprocessing applications for message queuing and for residence of online applications data

- In online, data base applications, such as management information systems

- In time sharing (or interactive) environments as swap devices and for online work storage (for program and data residence)

- As high-speed work storage for sorting, assembling, and link editing

- For residence of data indexes, such as for ISAM data sets

- As external page storage in a virtual storage environment

The 2305 Model 2 facility offers significantly faster access than, and almost three times the capacity of, the 2303 drum. In large Model 145 OS installations, the 2305 facility will be of benefit:

- As the primary system residence device

- In time sharing environments as a swap device and for program and data residence

- As high-speed work storage and for residence of data indexes

- As external page storage in a virtual storage environment

Summary

The combination of new and enhanced hardware, availability, and input/output facilities, expanded operating system support, integrated 1400/7010 emulation, and DOS emulation under OS provided by the Model 145 offers Model 30 and 40 users expanded computing capabilities without the necessity of a large conversion effort. Little or no time need be spent modifying operational System/360 code or programs currently being emulated. Users of 1400-series and 7010 systems can upgrade directly to a Model 145 and an operating system environment with a minimum of reprogramming, and DOS users can convert to OS more easily because of the availability of DOS emulation. Existing CPU-bound System/360 programs can execute faster because of the increased internal performance of the Model 145, while I/O-bound programs can benefit from the use of more storage, more channels, faster I/O devices, and block multiplexing.

The increased power and new functions of the Model 145 provide the base for expanded applications installation and penetration of previously marginal application areas. New applications installation and transition to more online operations can be easier when a virtual storage environment is implemented. The improved price performance of the Model 145 offers the user the opportunity to widen his data processing base at a lower cost than was previously possible.

SECTION 10: ARCHITECTURE DESIGN, SYSTEM TECHNOLOGY, AND SYSTEM COMPONENTS

10:05 ARCHITECTURE DESIGN AND SYSTEM TECHNOLOGY

ARCHITECTURE DESIGN

The basic design objectives embodied in the System/370 Model 145 provide System/360, 1401/1440/1460, and 1410/7010 users with a growth system in the intermediate system range that incorporates improvements and additions to System/360 architecture. The Model 145 provides significant new functional capabilities, performance improvements, and features to enhance system reliability, availability, and serviceability. This has been achieved under the following conditions:

- System/370 architecture is upward compatible with that of System/360 so that most user programs written for System/360 will run efficiently on the Model 145 without modification.

- Programming systems support for the Model 145 is based on certain programming currently provided for System/360 models, namely, OS (MFT and MVT) and DOS Version 3.

- Most currently announced System/360 I/O devices will operate on the Model 145. (See Section 20:05 for a list of the I/O devices that are not included in a standard Model 145 configuration.)

- The open-ended design characteristic of System/360 has been preserved and extended on System/370.

Extended System/370 architecture embodies two different modes of system operation, basic control mode and extended control mode, as determined by bit 12 of the current PSW. When a System/370 model 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 essentially is the System/360-compatible mode of System/370 operation.

When EC mode is operative in the Model 145, 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 removal of certain fields to create space for additional mode and mask bits that are required for new System/370 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 if 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 145 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 mode 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 704 bytes. This area can be reduced to 512 bytes by moving the CPU extended logout area. (See Section 50.)

5. Programs deliberately written to cause certain program checks.

6. Programs that depend on devices or facilities not implemented in the Model 145.

7. Programs that use model-dependent operations of the System/370 Model 145 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.

Note that these are the same types of implementation-dependent restrictions that exist for compatibility among System/360 models.

OS and DOS control programs are designed to support either BC or EC mode of system operation. Support of Model 145 systems operating in BC mode is provided by DOS Versions 3 and 4 and by OS MFT and MVT, each of which is extended to support certain new System/370 hardware features and I/O devices. Existing DOS Version 3, OS MFT, and OS MVT control programs generated for System/360 models can also be executed on a Model 145 operating in BC mode, if necessary (discussed in Section 60:30).

Support of Model 145 systems operating in EC mode is provided by DOS/VS, OS/VS1, OS/VS2, and VM/370, each of which is designated as system control programming (SCP). All of these operating systems support a virtual storage environment using dynamic address translation, which operates only when the system is in EC mode. VM/370 supports a virtual machine environment. User-written processing programs that operate on a Model 145 in BC mode, under DOS Version 3 or 4, OS MFT, or OS MVT, can be used with DOS/VS, OS/VS1, or OS/VS2, 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.

SYSTEM TECHNOLOGY

The Model 145 uses monolithic system technology (MST) for logic circuitry, as do other System/370 models. In addition, the Model 145 embodies a significant technological advance in the area of system storage implementation. That is, processor storage, as well as control and local storage, is implemented using monolithic technology instead of wired, discrete ferrite cores. The Model 145 is the first IBM system to use monolithic storage exclusively.

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 10.05.1, 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 component, and several chips and resistors were needed to form a circuit. In general, an SLT substrate contained a single circuit.

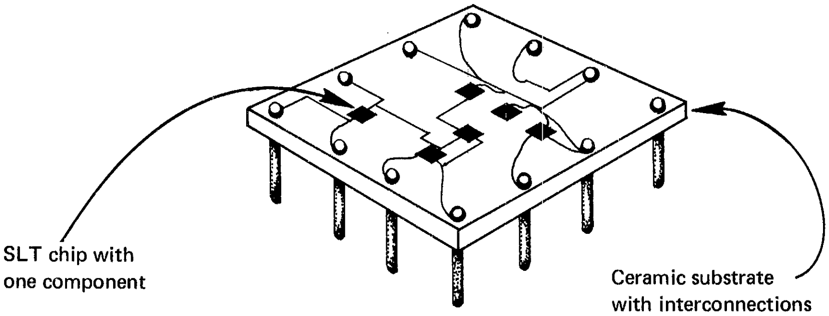

SLT chip with one component
Ceramic substrate with interconnections

Figure 10.05.1. 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. In the Model 145, an MST logic chip is slightly over a sixteenth of an inch square and contains over 100 components, which can form up to eight interconnected circuits. This compares to a single component on an SLT chip. 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. For example, the MST circuits in the Model 145 are about twice as fast as the SLT circuits in the Model 40.
- 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 chip. In the Model 145, a monolithic storage array chip is approximately an eighth of an inch square and contains a little more than 1400 components, or about 174 interconnected circuits. These circuits form storage bits and support circuitry on the chip. In the Model 145, one monolithic storage array chip contains 128 storage bits and their associated decoding, addressing, and sensing circuitry.

As shown in Figure 10.05.2, two storage array chips are mounted on a half-inch-square substrate, and a pair of substrates is packaged into a storage array module. Each half-inch-square storage array module, which contains 512 storage bits, is mounted on a storage array card, shown in Figure 10.05.3. The card is about 3-1/2 by 4-3/4 inches and contains 12K (12,288) storage bits. Storage array cards are placed in storage array boards which are called basic storage modules (BSM's). In outward appearance, therefore, monolithic storage resembles monolithic logic circuitry.

The primary basic storage module of the Model 145, which contains 48K bytes of storage (on 12K bit storage array cards) and its associated circuitry, is shown in Figure 10.05.4. It is approximately 13-1/4 inches long, 5-1/2 inches deep, and 9 inches wide. A 256K Model 145 contains six of these 48K-byte basic storage modules (256K of processor storage and 32K of control storage). When more than 256K of processor storage is installed, a main storage frame is required and 32K-byte BSM's are used in the main storage frame. The storage array cards used in a 32K-byte BSM contain 8K bits.

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 prior to 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 cards, each of which is a functional storage component. Diagnostic routines can be written that need only identify the failing storage card, which can be replaced in a matter of minutes. Storage increments can also be field-installed rapidly.

- 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. For example, 256K of core storage on the Model 40 requires almost twice the amount of space that 256K of monolithic processor storage requires on the Model 145.

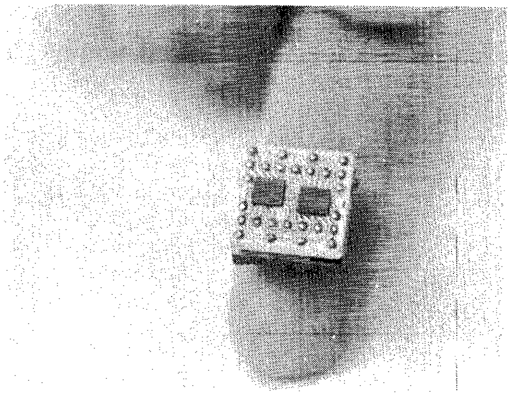


Figure 10.05.2. Monolithic storage array module containing 512 bits

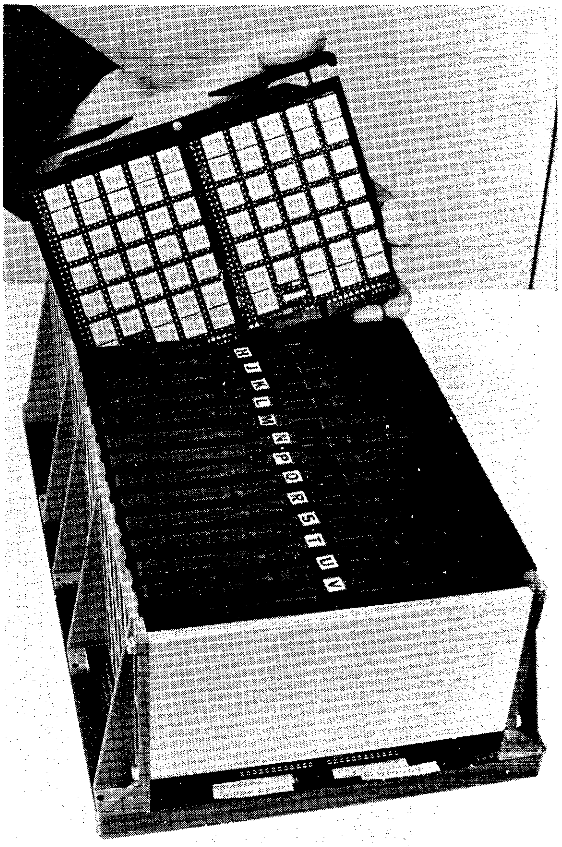


Figure 10.05.4. Basic storage module of the Model 145 containing 48K bytes of monolithic storage

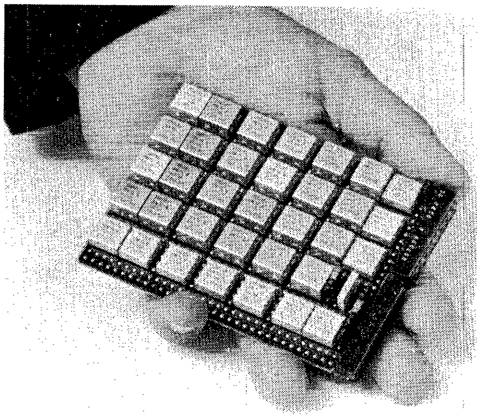


Figure 10.05.3. Monolithic storage array card containing 12K bits

MAJOR COMPONENTS

The major components of the Model 145 computing system are the processor (CPU), storage, channels, system control panel and console, and console file. Each component and its new features are discussed in the subsections that follow. Programming systems support of new BC mode features is covered in Section 30. Support of EC mode features is discussed in the optional programming systems supplements, which can be included as Sections 80 to 110. Reliability, availability, and serviceability (RAS) hardware features are mentioned only briefly. A full discussion of both hardware and programming systems RAS facilities is contained in Section 50.

10:10 THE CENTRAL PROCESSING UNIT (CPU)

The central processing unit contains all the elements necessary to decode and execute the instructions in the System/370 Model 145 instruction set and, optionally, those in the compatibility features required by the 1401/1440/1460 and 1410/7010 emulator programs. All CPU functions and channel operations are controlled by the microprogram contained in the 36-bit (4-byte) control words in reloadable control storage (RCS), which is housed in the CPU (discussed in Section 10:15).

The Model 145 has a variable-length CPU cycle time. Cycle times of 202.5, 247.5, 292.5, and 315 nanoseconds are implemented. The time required for the CPU to perform operations is made up of combinations of these cycles. The CPU fetches instructions from processor storage a doubleword at a time, while data accesses, both fetches and stores, are made on a word basis. Eight instruction bytes or four data bytes can be fetched by the CPU in 540 nanoseconds.

A single monolithic local storage is used as an intermediate storage area and is shared by the CPU and the integrated channels. Local storage contains the general and floating-point registers, subchannel information for the high-speed channels, and other items such as CPU working areas and multiplexer channel working areas. Two local storage sources can be accessed simultaneously.

The CPU also contains a set of monolithic register stacks, called external registers, that increase internal performance. These registers are a part of the data flow organization that is shared by both the CPU and the channels. Data for several operations can be maintained in these registers, thus eliminating temporary stores and fetches when control is switched from a channel to the CPU.

Extensive parity checking is done in the CPU to ensure the validity of the data being used. Every data path within and to the CPU is parity-checked, as are every microcode word and all adder sums. Automatic hardware retry of failing CPU operations, without programming assistance, is provided as an availability feature and is discussed in the RAS section.

Dynamic address translation and channel indirect data addressing are discussed in Section 15. Other significant new features of the Model 145 CPU are as follows.

CONTROL REGISTERS

The program states in which the Model 145 is operating are reflected in the current program status word (PSW) and in new CPU status indicators called control registers, which are located in the CPU. Up to 16 control registers, 0-15, can be addressed. Certain control registers are used only when EC mode is in effect. Control registers are program addressable when the CPU is in the supervisor state. A control register can be set with the new LOAD CONTROL instruction, and its contents can be placed in processor storage with the new STORE CONTROL instruction. Additional status indicators contained in control registers are required in order to support new system functions. A control register is 32 bits in size.

BASIC CONTROL MODE

As indicated previously, the contents, layout, and function of fixed locations 0-127 in System/370 models operating in BC mode are identical to these locations in most System/360 models with the exception of bit 12 in the PSW, which specifies EBCDIC or ASCII mode in System/360 models, and which specifies BC or EC mode in System/370 models. ASCII mode is not implemented in the Model 145, nor was the mode bit supported by IBM programming systems provided for System/360 models, as System/360 USASCII-8 did not become the ASCII standard. However, ASCII-encoded tapes will be supported by certain OS and DOS programs. That is, ASCII-mode tapes will be accepted as input and converted to EBCDIC for processing. The capability of writing ASCII-mode tapes is also provided. (See Section 30 for a discussion of OS MFT and MVT and DOS Versions 3 and 4 support of ASCII-mode tapes.)

To enhance system availability and serviceability, implementation of the machine check class of interruption in the Model 145 has been considerably altered from its implementation in Models 30 and 40 (see Section 50). However, the other four interruption classes operate in the same manner on Models 30, 40, and 145 except for the expansion of external interruption masking in the Model 145. Three external subclass mask bits, which allow selective masking of external signals (2-7), timing facilities, and console panel interrupt key interruptions, are provided in control register 0. When the PSW external mask bit is off, the CPU is disabled for all three external interruption types. When the PSW external mask bit is on, a console interrupt key, an interval timer, a time of day clock, a clock comparator, a CPU timer, or an external signal interruption occurs only if its associated subclass mask bit is on also.

EXTENDED CONTROL MODE

Extended control mode, unlike basic control mode, is exclusively a System/370 mode. 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.

Change in PSW Format

When a System/370 operates in EC mode, the format of the PSW differs from its BC mode format. Both PSW formats are shown in Figure 10.10.1. 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.

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 10.10.2. The balance of the fixed area for the Model 145, that which has model-dependent fields, is shown in Figure 50.10.3. The model-dependent area is not affected by whether BC or EC mode is specified, except for locations 185 to 187, which contain the I/O address for an I/O interruption and an IPL only when the Model 145 is in EC mode.

The machine check interruption procedure and the format of the data logged on a machine check (discussed in Section 50) are the same in EC and BC modes, except for differences in the PSW format and the permanently assigned locations previously discussed.

BC MODE PSW FORMAT

Bit	Content	
0	Channel 0 mask	System mask
1	Channel 1 mask	
2	Channel 2 mask	
3	Channel 3 mask	
4	Channel 4 mask	
5	Channel 5 mask	
6	I/O mask	
7	External mask	
8	Protect key	
9		
10		
11		
12	EC/BC mode (0 is BC)	
13	Machine check mask	
14	Wait/running state	
15	Problem/supervisor state	
16	Interruption code	
17		
18		
19		
20		
21		
22		
23		
24		
30		
31		
32	Instruction length code	
33		
34	Condition code	
35		
36	Program mask	
37		
38		
39		
40	Instruction address	
41		
42		
61		
62		
63		

EC MODE PSW FORMAT

Bit	Content	
0	0	System mask
1	PER mask	
2	0	
3	0	
4	0	
5	Translation mode (DAT feature mask)	
6	I/O summary mask	
7	External summary mask	
8	Protect key	
9		
10		
11		
12	EC/BC mode (1 is EC)	
13	Machine check mask	
14	Wait/running state	
15	Problem/supervisor state	
16	0	
17	0	
18	Condition code	
19		
20	Program mask	
21		
22		
23		
24	0	
30		
31		
32	0	
33		
34		
35		
36		
37		
38		
39		
40	Instruction address	
41		
42		
61		
62		
63		

Figure 10.10.1. BC and EC mode PSW formats

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 10.10.2. Model 145 model-independent fixed storage locations for BC and EC modes

Channel Masking Changes

When a System/370 operates in EC mode, interruptions from each channel are controlled by the summary I/O mask bit (bit 6) in the current PSW and an individual channel mask bit in control register 2. In the Model 145, bits 0 to 4 in control register 2 are assigned to control channels 0 to 4, respectively. Both the summary mask bit and the appropriate individual channel mask bit must be on in order for an interruption from a given channel to occur. In BC mode, interruptions from channels 0 to 4 are controlled by only the channel mask bits (bits 0 to 4) in the current PSW.

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 when dynamic address translation is present. The two additional bits (reference and change) are included for use with dynamic address translation and are discussed in Section 15:10. The SET STORAGE KEY instruction sets a seven-bit key regardless of the mode, BC or EC, in effect. The INSERT STORAGE KEY instruction causes a five-bit or a seven-bit key to be loaded into the register indicated, depending on whether BC or EC mode, respectively, is in effect.

Changes to Certain System/370 Instruction Definitions

All Model 145 instructions are valid in BC and EC modes. However, because of the 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)

INSERT STORAGE KEY

LOAD PSW

SET PROGRAM MASK

SET STORAGE KEY

SET SYSTEM MASK

SUPERVISOR CALL

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 in order 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 so modified because the interruption for SSM instructions and an SSM simulation routine, described next, are supported.)

Programs that use the other instructions listed do not have to be changed in order 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 in order to operate correctly in EC mode.

Program Interruption for a Set System Mask Instruction

When a Model 145 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 on, an attempt to execute an SSM instruction causes a program interruption without execution of the SSM instruction. When the SSM mask bit is off, 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 SSM interruption 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.

Program Event Recording

Program event recording (PER), a standard feature on the Model 145, 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 a one. 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 respective 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 the identification of the type of event is stored in the fixed 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. In the Model 145, instruction processing time is increased when the PER facility is operative.

If dynamic address translation mode is also specified when PER is active, virtual storage addresses instead of real storage addresses (discussed in Section 15) are placed in the control registers to monitor references to a contiguous virtual storage area.

EXPANDED INSTRUCTION SET

The standard instruction set for the System/370 Model 145 is a superset of that provided for System/360 Models 30 and 40. It consists of the System/360 instruction set plus new instructions that support System/370 architecture and provide additional functions. The Model 145 standard instruction set includes all general purpose and I/O instructions and all binary and decimal arithmetic instructions. Storage protect and time of day clock instructions are also standard. The new STORE CPU ID instruction permits a program to determine the model upon which it is operating and provides the system serial number. The new STORE CHANNEL ID instruction can be used to identify the types of channels present in the system. Other new instructions are:

- General purpose instructions

  Several general purpose instructions, which can be of benefit to both control and processing program performance, have been added to the Model 145 standard instruction set.

  SHIFT AND ROUND DECIMAL provides right or left shifting of packed decimal data using a single instruction. This instruction can save 6 to 18 bytes of instruction storage and instruction execution time for each decimal shift and round operation performed in commercial processing.

  MOVE LONG provides for the movement of up to 16 million bytes from one location in storage to another with a single instruction, thereby removing the current limitation of 256 bytes per move. A check for the possibility of destructive overlap is made by the hardware prior to the movement of any data and the MOVE LONG instruction is not executed if operand destruction can occur. This instruction can eliminate the necessity of multiple move instructions or the inclusion of move subroutines. The format and operation of MOVE LONG facilitates efficient record blocking and deblocking, field padding, and storage clearing, which are operations frequently performed in commercial processing. The new COMPARE LOGICAL LONG instruction can be used to compare logically two fields of up to 16 million bytes in length, thus removing the current 256-byte limit on byte compares. In addition, when an unequal compare occurs, the two characters that caused the inequality are identified.

  The MOVE LONG and COMPARE LOGICAL LONG instructions are interruptible. Thus, when an I/O operation terminates during their execution, the interruption is taken and the channel is not held up awaiting termination of what might be a lengthy move or compare.

  COMPARE LOGICAL, INSERT, and STORE CHARACTERS UNDER MASK instructions provide byte addressibility within the general registers and permit nonword-size data that is not on a word boundary to be compared with data in a register, loaded into a register, and stored from a register. These three instructions can be of most benefit to control program programmers, to compiler writers, and to others who must manipulate processor storage addresses.

  STORE THEN AND SYSTEM MASK and STORE THEN OR SYSTEM MASK are two 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.

- Extended Precision Floating Point

  The optional floating-point arithmetic feature includes floating point and extended precision floating-point instructions. Extended precision is provided for use in application areas in which the precision provided by the long-form floating-point format is not large enough.

  Precision of up to 28 hexadecimal digits, equal to up to 34 decimal digits, is provided by the extended precision data format. Extended precision is achieved by using two doublewords (16 bytes) to represent an extended precision floating-point number instead of using one doubleword as is done in long form representation. Fourteen hexadecimal digits, or up to 17 decimal digits, of precision are provided by the long floating-point format.

  Seven extended precision floating-point instructions are included in the optional floating-point arithmetic feature. They provide addition, subtraction, and multiplication operations for extended precision data, using a pair of floating-point registers, and the ability to round from long to short form or from extended to long form. An extended precision divide instruction is not provided; however, a simulator for this operation is provided in OS (discussed in Section 30).

BYTE-ORIENTED OPERANDS

The Model 145 supports a byte boundary alignment facility for processor storage. The presence of the byte-oriented operand function allows the processor storage operands of unprivileged instructions (RX and RS formats) to appear on any byte boundary without causing a specification program interruption. Without this facility, operands must be aligned on integral boundaries, that is, on storage addresses that are integral multiples of operand lengths. Byte orientation is standard and does not apply to alignment of instructions or channel command words (CCW's).

Use of byte alignment in a program degrades instruction execution performance. However, byte orientation can be used effectively in commercial processing to eliminate the padding bytes added within records or to blocked records to ensure binary and floating-point field alignment. The smaller physical record that results from the elimination of padding bytes requires less external storage and increases effective I/O data rates. I/O-bound commerical programs, in which throughput is in almost direct proportion to the I/O data rate, can achieve performance improvement by using byte alignment for binary and floating-point data.

A program written to use byte boundary alignment will not necessarily run on a System/360 model that does not have the feature. Therefore, programs that are to run on both the Model 145 and on System/360 models without byte orientation should be written to adhere to integral boundary rules.

MONITORING FEATURE

The monitoring feature is standard on the Model 145. This feature provides the capability of monitoring the occurrence of programmed events. For example, monitoring can be used to perform measurement functions (how many times a routine was executed) or for tracing functions for the purpose of program debugging (which routines were executed).

The MONITOR CALL instruction is provided with the monitoring feature. Execution of this instruction indicates the occurrence of one of the events being monitored. The operands of the MONITOR CALL instruction permit specification of up to 16 classes of events, each class with up to 16 million unique types of events. The 16 monitor classes are individually maskable via mask bits in control register 8. When a MONITOR CALL instruction is executed, a program interruption occurs, if the monitor class indicated is specified, and the event identification (class and type) is stored in the lower fixed storage area.

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 codes each instead of four events), determines when the events occur, and causes the program interruption by issuing MONITOR CALL instructions.

ARCHITECTURE IMPLEMENTATION ALTERATIONS

Two alterations have been made to the system action taken on a Model 145 during the execution of certain instructions common to both System/370 and System/360 models. The first involves all instructions that check the validity of operands involved in packed decimal operations. On the Model 145, an invalid sign in an operand causes the instruction to be suppressed (never executed) rather than terminated during execution as is done on System/360 models. Suppression, rather than termination, of an instruction when an invalid sign occurs ensures that the data fields involved remain unchanged. Therefore, a routine that inspects the field that has the invalid sign can be executed when a program check occurs. For example, when an invalid sign results from packing an entirely blank field, the sign can be corrected by programming, and transaction deletion or program termination is avoided.

The second alteration concerns the recognition of a storage protection exception during the execution of an EDIT or an EDIT AND MARK instruction. On a Model 145 a protection exception always occurs when a pattern character is fetched from a location protected for storing but remains unchanged during the edit operation. This change eliminates unpredictable system operation during editing on a Model 145. The occurrence of a protection exception for the situation described is model dependent for System/360 models.

INTERVAL TIMER

The interval timer at decimal location 80 in the fixed processor storage area is a standard feature and has a resolution of 3.33 ms instead of the 16.6-ms resolution implemented for the timer on Models 30 and 40. Its maximum time period remains 15.5 hours. The higher resolution of this interval timer eliminates many of the problems encountered in accounting routine accuracy caused by task execution durations that are shorter than the 16.6-ms resolution interval.

TIME OF DAY CLOCK

This clock is a binary counter of 52 bits with a cycle of approximately 143 years. It is a standard feature. The clock is updated every microsecond. Two new instructions (SET CLOCK and STORE CLOCK) are provided to set the time and to request that the current time be stored in the specified doubleword of processor storage. The time can be set only when the CPU is in supervisor state and only when the clock security switch on the system console panel is held in the enable set position.

The time of day clock can be used for more accurate time stamping than the interval timer. More accurate time of day can be maintained because, during normal system operation, the clock stops only when CPU power is turned off. (CE use of certain system test modes and an error in the clock will invalidate the clock time.) The interval timer cannot be as accurate as the clock for time of day maintenance because it is not updated when the system is in the stopped state and its updating may be omitted under certain conditions of excessive system activity. The 15.5-hour cycle time of the interval timer is also a restriction. The time of day clock better answers the timing needs of teleprocessing and real-time applications.

CLOCK COMPARATOR AND CPU TIMER

These timing facilities are an optional feature on the Model 145. 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 is implemented in control storage and has the same format as the time of day clock. The clock comparator is set to zero during IPL. 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 instead of the interval timer at location 80 to cause an interruption when a specified time is passed 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 format eliminates having to convert doubleword time of day clock values to single word interval timer values.

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 in local storage 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 such 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 retry operations) and while the CPU is in the wait state. The CPU timer 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 can be more 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 used 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 is reduced. This results 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 such that little of the processing currently required during a task switch, when the interval timer is used, is necessary.

10:15 STORAGE AND THE CONSOLE FILE

CONTROL AND PROCESSOR (MAIN) STORAGE

A significant new storage feature of the Model 145 is reloadable control storage (RCS) for microprogram residence. The use of writable storage for control functions adds to the advantages of using a read-only storage instead of conventional circuitry.

As implemented in the Model 145, use of RCS instead of read-only storage results in system cost savings and provides improved serviceability and additional system functions:

- System cost savings result because the total amount of control storage required is reduced. Fixed control storage addresses for each specific microcode function are not required. Since control storage is reloadable, the microcode required to support the hardware features of a specific system configuration can be efficiently packed in available control storage when the given system microcode is customized. In addition, all microcode for a system need not be resident at all times. For example, different versions of microcode for a given system containing different features can be loaded when required, and diagnostics overlay normal system microcode when they are needed. Furthermore, the fact that a single writable storage can be and is used for both control and processor storage helps achieve the price performance goal of the Model 145.

- Serviceability is enhanced because of the speed and ease of engineering change installation--the new microcode need only be loaded into RCS--and because more extensive diagnostics can be provided without the necessity of additional control storage (all control storage is available for diagnostic residence). The design simplicity of a single-storage system (one storage addressing design, a single set of sensing circuits, a common data flow design, etc.) also benefits serviceability and reduces system cost.

- Functional capability is extended by the ability to more easily support different architectures and features in one system. IBM-supplied 1400/7010 and DOS emulator microcode and special features are quickly and easily loaded.

A single monolithic storage, with a fetch cycle time of 540 nanoseconds and a store cycle time of 607.5 nanoseconds for four data bytes, is used for both processor and control storage in the Model 145. That is, the total storage present in the system (excluding local storage) is functionally divided into control and processor storage. Model 145 storage is available in the sizes shown below (K=1024 bytes).

Model*	Processor Storage	Control Storage	Total Storage
GE	160K	32K	192K
GFD	208K	32K	240K
H	256K	32K	288K
HG**	384K	32K	416K
I***	512K	32K	544K

*Model FED (112K) has been withdrawn
**Requires 3345 Main Storage Frame Model 1 or 3345 Storage and Control Frame Model 4 (128K)
***Requires 3345 Main Storage Frame Model 2 or 3345 Storage and Control Frame Model 5 (256K)

Total storage of up to 288K is housed in the CPU. A 3345 storage frame must be added to the system when more than 288K is present, as shown in Figure 10.15.1. Storage contained in the additional frame is addressed beginning with processor storage address zero.

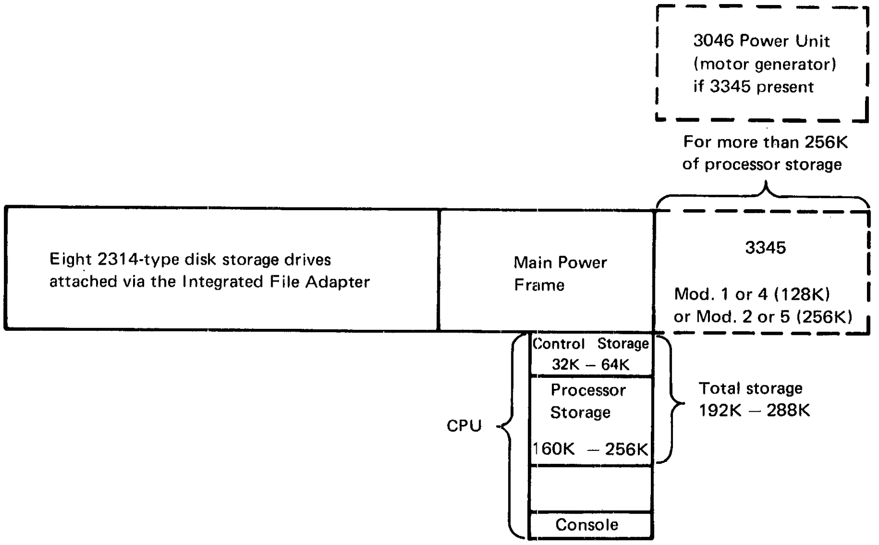

3046 Power Unit (motor generator) if 3345 present
For more than 256K of processor storage
Eight 2314-type disk storage drives attached via the Integrated File Adapter
Main Power Frame
3345
Mod. 1 or 4 (128K) or Mod. 2 or 5 (256K)
Control Storage 32K – 64K
Processor Storage
160K – 256K
CPU
Total storage 192K – 288K
Console

Figure 10.15.1. Model 145 physical layout

Models 1 and 4 of the 3345 contain 128K of monolithic storage (four 32K BSM's). Models 2 and 5 of the 3345 contain 256K (eight 32K BSM's). When a 3345 Model 1, 2, 4, or 5 is present in a Model 145 configuration, the IBM 3046 Power Unit, which contains a motor generator (MG) set, is also required to supply 400 Hz (cycle power) for the storage in the 3345 unit. Models 3, 4, and 5 of the 3345 contain integrated storage control for attachment of 3330-series disk storage and are discussed in Section 20:10.

Control storage is always assigned the high-order range of available storage addresses and is always a minimum of 32K. The processor/control storage boundary within total storage is determined by the value in the address check boundary register in the CPU. No microcode can be executed until this boundary is established. The addresses below this register value are processor storage addresses, while those equal to and above it are control storage addresses. An attempt by a program instruction to reference an address equal to or above the boundary value results in an address check program interruption. A control storage access by the microprogram below the boundary value results in a machine check. Thus, in a system containing total storage of 192K, the boundary would be set at a value of 163,840 in order to provide 160K of processor and 32K of control storage. The boundary value is set when microcode is loaded (discussed under "The Console File" at the end of this subsection).

The total amount of control storage required in a configuration is dependent upon the features present in the system. Control storage contains system microcode and byte and block multiplexer channel unit control words (UCW's). A movable control storage boundary facility is implemented so that when more than 32K is required, control storage can be expanded in 2K (2048-byte) increments up to a maximum of 64K at the expense of processor storage. This is done by lowering the value set in the address check boundary register.

The basic system microcode (of 26K) includes all the microcode required by the standard instructions and features of the Model 145 (including that for the byte multiplexer channel but excluding that for its standard 16 UCW's) in addition to a microcode patch routine and area (discussed under "The Console File" at the end of this subsection). Additional control storage is required to contain each of the following selectable items:

- Byte and block multiplexer UCW's
- Block multiplexer mode for selector channels
- Console printer-keyboard
- Integrated file adapter
- 1401/1440/1460 compatibility
- 1401/1440/1460 and 1410/7010 compatibility
- Floating-point arithmetic
- Direct Control
- CPU timer and clock comparator

Table 10.15.1 shows the total control storage requirements for sample configurations. (See <u>IBM System/370 Model 145 Functional Characteristics</u>, GA24-3557, for specific control storage requirements by function.) If disk cartridges that contain different feature mixes are desired by an installation, they can be requested from IBM via a special order.

Error checking and correction (ECC) hardware provides automatic detection and correction of all single-bit processor and control storage errors, and detection, but not correction, of all double-bit and most multiple-bit errors. The ECC feature is discussed fully in the RAS section.

Table 10.15.1. Total control storage requirements for sample Model 145 hardware configurations. All support DOS emulation under OS as well.

Configuration		
1	2	3
Basic system (includes selector channel support) 16 byte multiplexer UCW's 3210-1 console 1401/1440/1460 compatibility	Basic system (includes selector channel support) 16 byte multiplexer UCW's 3210-1 console Integrated File Adapter	Basic system (includes selector channel support) 128 byte multiplexer UCW's 3215 console and remote 3210-2 Block multiplexer mode Block multiplexer UCW groups - 4 (32 UCW's)
34,656 bytes (34K)	39,216 bytes (40K)	35,374 bytes (36K)
(without 1401/1440/1460 compatibility, 29,456 bytes total)	(with 1401/1440/1460 compatibility, 44,416 bytes total)	(with floating point and 1401/1440/1460 compatibility, 42,814 bytes total)

THE CONSOLE FILE

Control storage is loaded directly from a small read-only disk device, called the console file, which is a basic component of the Model 145. This file is located beneath the operator's console table, and it reads removable, prerecorded disk cartridges, several of which can be stored near the console file. A disk cartridge has a capacity of 65,280 ten-bit bytes (eight data bits, one parity bit, one start bit), and a data track is divided into eight sectors for recording purposes. Data is read from the disk cartridge by the file at a rate of 33,300 bits per second.

The disk cartridge is contained in a protective eight-inch-square casing, as shown in Figure 10.15.2. When mounted on the console file, the cartridge rotates inside its casing, and data is read through a hole in the casing that exposes the data recording area. Reading from the console file is initiated by console switches and buttons. Once the file has been started, its operation is controlled by command bytes that are interspersed within the data (microcode or diagnostics) contained on the disk cartridge tracks. There are no I/O instructions or commands that a program can execute to cause reading from the file, and there is no way for any installation to write data on a disk cartridge.

Prewritten disk cartridges containing all the microcode required for the specific Model 145 configuration are shipped to each Model 145 installation. One standard and one backup cartridge containing the customized system microcode are provided in addition to several other disks that contain system diagnostics. The processor/control storage boundary address is contained on the customized disk cartridge sent to the installation and reflects the amount of control storage required for the configuration. The only time the address check boundary register can be set is during microprogram loading from the disk cartridge. A single cartridge has enough capacity to contain all the available system microcode and to establish the maximum number of byte and block multiplexer UCW's in control storage.

In order to load control storage with the system microcode, the proper disk cartridge must be mounted on the console file, the diagnostic/file control switch on the system console panel must be in the process position, and the CPU must be in process mode. When the system is in this status, an initial microprogram load (IMPL) is initiated automatically when system power is turned on.

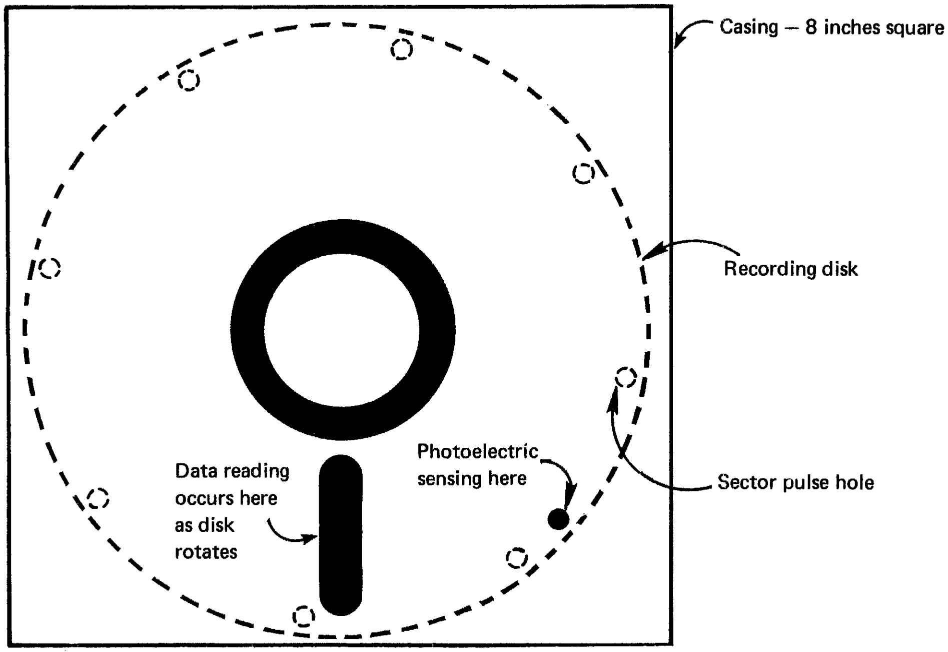

Casing – 8 inches square
Recording disk
Sector pulse hole
Photoelectric sensing here
Data reading occurs here as disk rotates

Figure 10.15.2. Disk cartridge for the console file. Shaded areas represent recording disk showing through cutout areas in casing. Dotted lines show recording disk inside the casing. Eight holes in the periphery of the disk generate, via photoelectric sensing, sector pulses to indicate beginning of sector.

When power is already on, IMPL is initiated via the start console file key. Pressing this key causes power to be turned on in the file (it is normally off), and as soon as the disk is up to speed--about five seconds--reading from the console file begins. The key remains red from the time it is pressed until reading begins and then turns white to indicate that loading has begun successfully. The key remains red if the disk does not turn, a disk is not mounted, etc.

Approximately 45 seconds are required for the loading of 32K of control storage (includes maximum arm positioning time). During loading, considerable error checking occurs to ensure that the microcode is read and loaded into control storage correctly. If an error is detected, the console file key turns red to inform the operator. The white indication and file power are turned off at the successful completion of control storage loading. 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.

The basic system microcode control storage space requirement includes room for a patch area, into which the user or the customer engineer can load temporary patch engineering changes (EC's) from an I/O device at the completion of IMPL. When temporary patches are received, the customer engineer plugs a jumper card into the system to cause a request for patch loading to be typed on the console at the completion of each IMPL procedure. Checking is performed to ensure that EC's are correctly loaded. Once microprogram patches have been made, no additional changes can be made until control storage is again loaded. EC's distributed as temporary patches will be included in the next customized system microcode disk cartridge sent to each Model 145 installation, at which time the customer engineer will remove the jumper card.

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 can be performed automatically, as discussed above). Normally, the cartridge containing the system microcode will stay mounted on the console file and cartridge changing will occur only when diagnostics are performed. A disk cartridge containing customized system microcode contains the serial number of the system for which it is customized and is not portable from one Model 145 system to another.

The console file is also used for loading and executing diagnostic routines, and it is a basic debugging tool for the system. A comprehensive set of fault-locating microdiagnostics is supplied to each installation on disk cartridges that can be loaded directly from the console file into the Model 145 and executed. These microdiagnostics are discussed in Section 50:15.

10:20 CHANNELS

The Model 145 user has a broad range of channel facilities from which to choose when configuring the input/output of a system. While channel functions compatible with those available on Models 30 and 40 are provided, the Model 145 offers new facilities, more varied configuration capabilities, and faster channel data rates. In addition, faster I/O devices can be attached. These capabilities enable the Model 145 user to tailor a system to his I/O processing needs, on an improved price performance basis, to increase channel throughput.

A byte multiplexer channel with 16 subchannels and one selector channel are standard. Optional features include additional byte multiplexer subchannels, an Integrated File Adapter for direct attachment of 2314A-type direct access storage devices, additional selector channels, block multiplexer mode of selector channel operation, and a channel buffering feature to increase selector channel data rates.

Model 145 channels are integrated. They share with the CPU the use of control storage, use of the CPU and processor storage data flow, and use of the CPU arithmetic logic unit. The byte multiplexer channel interferes with CPU operations when it requires one of the shared components. Selector channels and the integrated file adapter interfere with both the CPU and the byte multiplexer channel. The channels interfere with CPU operations when an I/O operation is started and when it completes, as well as when they require control or processor storage cycles.

Comprehensive error checking has been incorporated in the basic design of the channel hardware. Checking is performed on the control logic in most areas, and standard parity checking is done on the data flow. Improved error recovery hardware has also been included (discussed fully in the RAS section).

The standard instruction set also includes a new I/O instruction, called HALT DEVICE. This instruction is specifically designed to stop an I/O operation on a particular device on a byte or block multiplexer channel without interfering with other I/O operations in progress on the channel. HALT DEVICE should always be used, instead of HALT I/O, to stop an I/O operation on a multiplexer channel.

The Channel-to-Channel Adapter feature available for System/360 models is also an optional feature for the Model 145. It allows two

System/370 channels or a System/370 and System/360 channel to be interconnected. Only one adapter can be installed on a Model 145.

BYTE MULTIPLEXER CHANNEL

The standard byte multiplexer channel provided for the Model 145 is functionally identical to the one available on System/360 Models 30, 40, and 50. It operates in byte interleaved mode, permitting several slow-speed devices to operate concurrently, or in burst mode, allowing one high-speed device to function.

Data is transferred between the byte multiplexer channel and processor storage one byte at a time. The maximum byte multiplexer channel data rate in byte mode is 50 KB and in burst mode is 180 KB. These rates allow for typical interface delays but do not include IFA, selector channel, or block multiplexer channel interference.

The byte multiplexer channel can have up to 256 subchannels. The number of subchannels in a system is not related to the size of processor storage, as it is on the Model 40. The number of subchannels desired (in addition to the standard 16 assumed) must be ordered by the user so that enough control storage for the UCW's is allocated and initialized by the customized installation microcode disk cartridge. Configurations of 16, 32, 64, 128, or 256 byte multiplexer subchannels are the only possibilities, and any configuration is permitted with any processor storage size.

The number of subchannels present determines the maximum number of concurrent I/O operations that can execute on the byte multiplexer channel. Each subchannel is associated with a unit control word (UCW). UCW's are used to store channel register data between data transfers to and from processor storage when devices are operating on the byte multiplexer channel. UCW's are contained in control storage. A byte multiplexer UCW is 16 bytes in length and the maximum number of byte multiplexer UCW's permitted (256) can be contained in 4K of control storage.

INTEGRATED FILE ADAPTER

One Integrated File Adapter (IFA) can be installed on a Model 145. This optional feature allows direct attachment to the Model 145 of from three to eight 2314A-type drives without the necessity of installing a selector channel and disk control unit. It provides the Model 145 with a lower cost method of attaching 2314 disk storage. In addition, power supply and space requirements are reduced.

The 2319 Disk Storage unit Model A1, which consists of three disk drives, must be the first unit attached to the Model 145 via the IFA. The disk drives in the 2319 are functionally compatible and program compatible with 2314A-type disk storage drives, and the interchangeable 2316 Disk Pack is used as the storage medium. The 2319 has a maximum capacity of 87 million bytes, up to 29 million bytes per drive. Up to five additional 2314A drives can be attached via the IFA. Any combination of 2312 (one drive), 2313 (four drives), and 2318 (two drives) Disk Storage, up to the maximum of five additional drives, is permitted with the 2319 Model A1 for a maximum facility capacity of 233 million bytes. Alternatively, the 2319 Model A2 and the 2312 or 2318 can be attached to the IFA with a 2319A1 to provide a six, seven, or eight drive facility. The 2319A2 is functionally equivalent to the 2319A1 but attaches to a Model 145 IFA only via the 2319A1.

Facilities consisting of up to nine 2314-type disk storage units can also be channel attached via 2314 A1 and B1 control units. The ninth drive can be used only as a spare. Listed below are the options available for integrated and channel attached 2314 disk storage.

	Facility	Unit	Number of Drives	Millions of Bytes	Notes
1.	IFA with 3 to 8 drives (87 to 233 million bytes)	2319A1	3	87	The 2319A1 must be first unit attached via the IFA. Any combination of 2312, 2318, 2313, up to 5 drives, can be added.
		2312	1	29	
		2318	2	58	
		2313	4	116	
2.	IFA with 3, 6, 7, or 8 drives (87 to 233 million bytes)	2319A1	3	87	The 2319A1 must be the first unit attached via the IFA. The 2319A2 must be attached via the 2319A1. Either one 2312 or one 2318 in addition to the above can be added.
		2319A2	3	87	
		2312	1	29	
		2318	2	58	
3.	Channel with 2314A1 control unit and 1 to to 9 drives (29 to 233 million bytes)	2312	1	29	Any combination up to nine drives (ninth must be a spare).
		2318	2	58	
		2313	4	116	
4.	Channel with 2314B1 control unit and 3, 6, or 9 drives (87, 174, or 233 million bytes)	2319B1	3	87	The 2319B1 must be the first unit attached to the 2314B1. The 2319B2 must be attached to the 2319B1.
		2319B2	3	87	
		2319B2	3	87	

The integrated file adapter is addressed as channel 1 and its drives are addressed as 130 to 137. If the IFA is present, selectors 1 and 4 cannot be installed on the Model 145 and the standard selector is addressed as channel 2. The IFA performs the functions of a selector channel and disk control unit for the disk drives attached to it and is programmed as though channel 1 and a control unit were present. The Record Overflow and File Scan features are standard on the IFA. However, 2844 Auxiliary Storage Control, the Channel Word Buffer, the Channel-to-Channel Adapter, block multiplexing, and the Two-Channel Switch features do not apply to the IFA. In addition, the IFA cannot handle data-chaining operations within the data area of a disk record.

SELECTOR CHANNELS

A single selector channel is standard on the Model 145. If the IFA is not installed, selector channel 1 is standard and, optionally, channels 2, 3, and 4 can be installed. If the IFA is present, selector channel 2 is standard and only channel 3 can be added. The basic system microcode includes all the microcode required to support any configuration of selector channels.

Selector channels on a Model 145 are functionally equivalent to System/360 selector channels but support significantly higher data rates than Model 30 and 40 channels. An individual selector channel without the optional Channel Word Buffer feature installed can sustain a data rate of .82 MB, while an aggregate channel data rate of 1.5 MB for the system is possible. Data is transferred between a selector channel and processor storage on a one-byte basis. A one-byte store into processor storage requires 607.5 nanoseconds, while a one-byte fetch requires 540 nanoseconds.

The data rate of a selector channel can be increased by installation of the optional Channel Word Buffer feature. When this feature is present, a four-byte buffer is included for each installed selector channel. (The Channel Word Buffer feature does not apply to the IFA or the byte multiplexer channel.) Data is transferred between a channel buffer and processor storage up to four bytes at a time to increase the channel data rate and reduce the number of required processor storage accesses, thereby causing less interference with the CPU. If the word buffer is installed, an individual selector channel can sustain a data rate of 1.85 MB, while the maximum obtainable aggregate data rate of all operating channels is approximately 5 MB. The transfer times stated for single-byte fetches and stores are not altered by use of the word buffer. The Channel Word Buffer feature is a prerequisite for attachment of the 2305 Model 2.

The order of priority implemented for handling simultaneous requests for an I/O interruption is (high to low): byte multiplexer channel, IFA or channel 1, channel 2, channel 3, channel 4. Simultaneous requests for a command- or a data-chaining operation are handled on a rotating basis in the following high-to-low priority sequence: IFA or channel 1, channel 2, channel 3, IFA or channel 1, etc. Selector channels and the IFA transfer data to and from processor storage during a share-cycle. The high-to-low priority sequence for servicing share-cycle requests is channel 2, IFA, channel 3, channel 2, etc., when the IFA is present, and channel 1, channel 2, channel 3, channel 4, channel 1, etc., when the IFA is not installed. The use of a rotating technique ensures that no channel has more than one request serviced when a service request is outstanding for another channel.

Tables 10.20.1 and 10.20.2 show examples of maximum-speed I/O configurations with currently announced I/O devices that will operate on the Model 145 without and with the Channel Word Buffer feature installed. The aggregate channel data rate listed is the rate that can be sustained at any instant without data overrun when all installed high-speed channels are operating concurrently.

BLOCK MULTIPLEXER CHANNELS

Block multiplexing mode for the Model 145 is an optional, no-charge feature. It permits any or all installed selector channels to operate as block multiplexer channels. When a system is ordered, the user specifies those channels that are to operate only as selector channels and those that are to have the capability of operating in either selector or block multiplexer mode. Block multiplexing will be of most benefit to system throughput when used with direct access devices with rotational position sensing capability, such as the 3330-series and the 2305. The block multiplexer feature must be installed if 3330-series or 2305 disk storage is attached to the Model 145, OS is used, and multiple requesting and rotational position sensing support are desired. (DOS does not support block multiplexing.)

The setting of a channel mode bit in a control register determines whether a channel with block multiplexing capability operates in block multiplexer or selector mode. Those channels designated as selector channels only are not affected by this mode bit. The mode bit is set to selector mode at IPL and can be altered by programming at any time. When a START I/O instruction is issued to a channel with block multiplexing capability on the Model 145, the current setting of the channel mode bit determines the mode in which the subchannel involved will operate.

Table 10.20.1. Sample Model 145 I/O configurations without the Channel Word Buffer feature installed

Channel	Configuration		
	1	2	3
IFA	2319 disk 312 KB	-	-
1	-	3330 disk 806 KB	*3420 M7 tape 320 KB
2	*3420 M7 tape 320 KB	3420 M7 tape 320 KB	*3420 M7 tape 320 KB
3	*3420 M7 tape 320 KB	-	*3420 M7 tape 320 KB
4	-	-	3420 M5 tape 200 KB
Aggregate rate	.95 MB	1.12 MB	1.16 MB
*Or 2314/2319 disk storage			
Note: When the Channel Word Buffer feature is not present, I/O operations on channels 3 and 4 are not possible when a 3330-series string is operating on channel 1.			

Table 10.20.2. Sample Model 145 I/O configurations with the Channel Word Buffer feature installed

Channel	Configuration		
	1	2	3
IFA	2319 disk 312 KB	-	-
1	-	2305 M2 disk 1.5 MB	3330 disk 806 KB
2	2305 M2 disk 1.5 MB	3330 disk 806 KB	3330 disk 806 KB
3	3330 disk 806 KB	3330 disk 806 KB	3330 disk 806 KB
4	-	3420 M7 tape 320 KB	3420 M7 tape 320 KB
Aggregate rate	2.62 MB	3.43 MB	2.74 MB
Note: I/O devices, such as direct access storage, that have a time dependency on command chaining will overrun if placed on channel 4 whether or not the Channel Word Buffer feature is present.			

The block multiplexer channel is designed to increase system throughput by increasing the amount of data entering and leaving the system in a given period of time (the effective data rate). Better use of channel time can be achieved by operating the channel in block multiplexing mode. A single block multiplexer channel can support interleaved, concurrent execution of multiple high-speed channel programs. The block multiplexer channel can be shared by multiple high-speed I/O devices operating concurrently, just as the byte multiplexer can be shared by multiple low-speed devices. Like the byte multiplexer, the block multiplexer channel has multiple subchannels, each of which has an associated UCW in control storage and can support one I/O operation.

Block Multiplexer Channel Operation

A block multiplexer channel functions differently from a selector channel in the way in which it handles command-chained channel programs. A selector channel or a block multiplexer channel operating in selector mode executing a command-chained channel program is busy during the entire time the channel program is in operation, whether data transfer is occurring or not. A block multiplexer channel executing a command-chained channel program has the ability to disconnect from the operational channel program during certain non-data transfer operations. That is, a block multiplexer channel can be freed during a nonproductive activity, for example, during disk seeking and most record positioning, thereby allowing more data to be transferred per unit of channel busy time.

Block multiplexing operates as follows. Assume a block multiplexer channel is executing a channel program consisting of multiple command-chained CCW's. When channel end is presented without concurrent device end, the channel disconnects from the I/O device and becomes available for an I/O operation on another device--even though the channel program of the disconnected device is not complete. At channel disconnect time the subchannel and the device's control unit retain the information necessary to restart the disconnected channel program.

When the device signals that it is again ready for the channel (by presenting device end), its control unit attempts to regain use of the channel. If the channel is free at this time, the channel registers are reloaded with the information previously saved (in the device's UCW), and the disconnected channel program is resumed at the appropriate CCW. If the channel is busy when reconnection is requested, the device must wait until it becomes available. Once multiple channel programs have been initiated on one channel, the interleaving of data transfer operations is controlled by block multiplexer channel hardware and the control units of the devices operating in block multiplexing mode.

To facilitate channel scheduling on block multiplexer channels, a new interruption condition, called channel available, has been defined. At disconnect time for a channel program, the block multiplexer channel is available for the resumption of an uncompleted channel program previously started, or another channel program can be initiated. A channel available interruption occurs at disconnect time to indicate channel availability if a START I/O, TEST I/O, TEST CHANNEL, or HALT DEVICE instruction was issued previously while the block multiplexer channel was busy.

Two additional facts should be noted about block multiplexer channel operations:

1. When multiple channel programs are operating concurrently in block multiplexing mode, a device can regain control of the channel only when the channel is not busy. Thus, only cyclic devices (such as direct access devices with rotational position sensing) or buffered devices (such as the 2540 Card Read Punch and the 1403 Printer) can disconnect during the execution of a command-chained channel program on a block multiplexer channel and resume operation later.

2. Data transfer operations for concurrently operating devices on a block multiplexer channel are interleaved on a first-come, first-served basis as the desired records become available. Thus, devices are serviced in the order in which their records become available, not necessarily in the order in which their channel programs are initiated.

Block Multiplexer UCW Assignment

The number of UCW's required for the block multiplexer channels installed on a Model 145 depends upon the number and types of I/O devices in the system configuration. When a system is ordered, the number of UCW's required must be specified so that a UCW address table is established and enough control storage is reserved for a pool of the specified UCW's by the customized disk cartridge sent to the installation. The maximum number of block multiplexer UCW's possible per system is 512 (assuming all are nonshared).

I/O devices are grouped into three types for the purpose of UCW assignment as follows:

- Type 1 - each device in the group requires a nonshared UCW
- Type 2 - all devices in the group use the same shared UCW
- Type 3 - a UCW is not assigned to any device in the group and the devices operate in selector mode (there is no disconnection during command-chained operations)

A UCW (or a subchannel) is referred to as nonshared if it is associated and can be used with only one device. Examples of Model 145 devices that should be assigned to a nonshared UCW because they have block multiplexing capability are:

- 3330-series disk storage - one UCW per drive in the string is required; a block of eight UCW's is always assigned.
- 2305 facilities - eight UCW's per 2305 module are required.
- 2540 Card Read Punch unit - one UCW for the reader and one for the punch
- 3505 Card Readers and 3525 Card Punches - one UCW for each reader and one for each punch
- 1403 Printers attached to a 2821 Control Unit - one UCW per printer
- 3211 Printers - one UCW per printer

A shared UCW can be used by a set of devices, one device at a time. A shared UCW generally is assigned to a control unit that has multiple devices attached, only one of which can be in operation at a time.

Devices for the Model 145 that must be assigned to a Type 3 group (cannot have a UCW assigned and must operate in selector mode) are magnetic tape units and direct access devices without rotational position sensing, such as the 2311, 2321, 2303, 2319, and 2314.

The type and number of I/O device groups in a given system configuration and the addresses to be assigned to the shared block multiplexer UCW's are established when a Model 145 is installed. The customer engineer inserts this information into a plug card for each channel and during the IMPL procedure the UCW address table is initialized with the data from the plug cards.

A nonshared block multiplexer UCW requires eight bytes of control storage. The nonshared block multiplexer UCW's in a Model 145 are not hard-wired to specific channels and are assigned in groups of eight, representing eight device addresses. System reset causes all nonshared UCW's to become unassigned and available for dynamic assignment to nonshared devices on any installed block multiplexer channel. When system operation begins, a nonshared UCW is assigned to an I/O device when the first START I/O instruction to the device is given, as described below. (A UCW is not assigned unless the device is actually present.)

When the first START I/O is executed for a nonshared device, the channel determines whether a block of eight UCW's has been assigned to the range of eight addresses in which the device falls, 190-197, 230-237, etc. If none are allocated, a block of eight available UCW's is assigned to the channel for the address range of that device. When the first START I/O is initiated to another I/O device on the same channel and in that block of addresses, the channel determines that a block of UCW's has already been assigned.

This process continues for all nonshared I/O devices until each device has a UCW assigned. If the pool of nonshared UCW's is empty when a request is made, a channel not operational condition code setting is given (condition code 3) and the START I/O is not executed. When a larger UCW pool is required, a new disk cartridge must be obtained from IBM in order to allocate more UCW's to the pool in control storage.

Shared block multiplexer UCW's are not dynamically assigned to devices as are the nonshared. The user must determine the number of shared UCW's required and the device addresses they are to be assigned so that this information can be placed in the plug cards. Each shared UCW requires one UCW area (8 bytes) plus one additional byte. Thus, a single UCW group of eight UCW's (64 bytes) provides seven shared UCW's.

The total number of UCW groups required in a system is equal to the number of shared control units in the I/O configuration divided by seven (rounded high), plus the number of UCW groups required by nonshared control units. The number of groups assigned to the pool must be a multiple of two.

For example, assume an I/O configuration includes the following:

- One 3830 control unit with two 3330-series modules (four drives) on block multiplexer channel 1
- One shared control unit with six devices on block multiplexer channel 2
- One shared control unit with six devices on block multiplexer channel 3

The 3330-series string requires assignment of one nonshared UCW group (eight UCW's), of which four UCW's will actually be used. Four more drives can be added to the 3330-series string. The devices on channel 2 require a single shared UCW, as do the devices on channel 3. Thus, one group of shared UCW's, of which only two UCW's will be used, suffices. Five more shared control units can be added to the system before the shared UCW group is exhausted. This I/O device configuration requires two UCW groups, one shared (Type 1) and one nonshared (Type 2), or 128 bytes of control storage. If an additional string of 3330-series drives is added to this configuration, a second nonshared UCW group is needed and two more UCW groups must be specified for inclusion in the system microcode (for a total control storage requirement of 256 bytes for block multiplexer UCW's).

Dynamic assignment of nonshared block multiplexer UCW's provides more flexibility in I/O configurations and requires allocation of less control storage than does fixed assignment of an equal number of UCW's to each block multiplexer channel. Only the requested number of UCW's, based on the specific I/O configuration of a system, occupies control storage.

10:25 BLOCK MULTIPLEXING OPERATIONS WITH ROTATIONAL POSITION SENSING DEVICES

Rotational position sensing (RPS) and multiple requesting are standard features for 3330-series and 2305 direct access storage devices. These two functions, together with block multiplexing, are designed to increase system throughput by increasing channel throughput.

The presence of RPS in the control unit of a direct access device enables it to operate in block multiplexing mode. The use of rotational position sensing reduces the number of channel programs that have to be initiated for direct access devices that require an arm-positioning seek (such as the 3330-series), frees channels more often during direct access device operations--specifically, during most of the time required to position a track to a desired record--and permits disk channel programs to be initiated sooner on block multiplexer channels than is possible with selector channels.

Multiple requesting is implemented in a direct access device control unit to enable it to handle concurrent execution of multiple RPS channel programs. Model 1 of 3830 Storage Control for 3330 modules, for example, can simultaneously control eight RPS channel programs, one on each of its drives.

In order to overlap seek operations for direct access devices without RPS, channel scheduling routines must initiate two channel programs for each record read or write. The first is a stand-alone seek, which frees the channel as soon as the control unit accepts the seek address. (The control unit is also free during arm movement.) At the completion of the seek, a device-end interruption is presented, and the data transfer channel program is subsequently initiated to search for the desired record and transfer the data. A selector channel is busy during the entire search operation (execution of the SEARCH command by the control unit) that locates the desired disk record on the track. Search time can be significantly greater than data transfer time for disk records smaller than half a track in size. Search time averages one-half of a rotation for a read or write (8.3 ms for a 3330-series drive) and requires a full rotation, less record write time, for a write verification chained from a write.

Use of RPS reduces the time the channel is busy during the search for a disk record. It permits the SEARCH command to be initiated just before the desired record is to come under the read/write heads, that is, when the desired rotational position is reached. To accomplish this, a "sector" concept is employed. The tracks in each cylinder of a direct access device are considered to consist of equally spaced sectors (the number of sectors varies by device). Track formatting is unchanged but each record has a sector location as well as a record address. A sector is not physically indicated on disk tracks, but is the length of the track arc that passes under the read/write heads in one sector time. For 3330-series drives, for example, sector time is defined to be approximately 130 microseconds. Thus, there are 128 sectors per track on 3330-series drives.

A disk control unit with RPS and multiple requesting can determine the sector currently under the heads of each of its drives. A sector counter is contained in each drive. The counter is incremented once every sector time period and set to zero each time the index marker passes under the heads. The sector in which a record falls is a function of the length of all records that precede it and of its sequential position on the track. Therefore, sector location can be calculated for fixed-length records.

Two new disk commands are provided for use with rotational position sensing: SET SECTOR and READ SECTOR. If the sector address of a record is known or can be calculated, a SET SECTOR command can be included in the disk channel program to cause the control unit to look for the designated sector. Once the control unit accepts the sector number provided by a SET SECTOR command, both the block multiplexer channel and the disk control unit disconnect and are available for another I/O operation. When SET SECTOR is used for positioning, the time the channel is busy during the search for a record is reduced from an average of 8.3 ms to an average of 260 microseconds for 3330-series drives. (Allowing for the worst case of speed variation and for disk pack interchange, the search time for a record, from sector found to beginning of desired record, can vary from 120 microseconds to 380 microseconds on a 3330-series drive.)

The READ SECTOR command is useful for sequential disk processing and for write verification. When chained from a READ, WRITE, or SEARCH command, READ SECTOR provides the sector number required to access the record processed by the previous CCW. This sector number can be used to reposition the track in order to verify the record just written or in order to read or write the next sequential record. These two new sector commands, used in conjunction with the block multiplexer channel, permit a single command-chained channel program, which frees the channel and disk control unit during seek and rotational positioning operations, to be initiated for each disk operation.

When record ID is known, the two channel programs shown below illustrate direct retrieval of a record from an OS BDAM data set on a direct access device without RPS, such as the 2314 (key not written). The seek operation can be overlapped with other seeks and one data transfer operation on the same selector channel. (Commands shown in the two channel program examples are only those that illustrate the advantage of RPS. Thus, commands such as SEEK HEAD and SET FILE MASK, which are used by data management to ensure correct operation, are not shown.)

<u>Channel program 1.</u> Initiate the stand-alone seek to position the disk arm.

Command Chaining Flag	Command	Selector Channel and Disk Control Unit Status
	SEEK (Seek address)	Free as soon as control unit accepts seek address

<u>Channel program 2.</u> Initiate the data transfer operation after the seek is complete.

Command Chaining Flag	Command	Selector Channel and Disk Control Unit Status
CC	SEARCH ID EQ (ID - sequential position on the track)	Busy (12.5 ms on average for 2314)
CC	TIC (Back to search if ID not equal)	
	READ DATA (Storage address of input area)	Busy

When the sector address is known or can be calculated, the channel program below illustrates retrival of a record from the same BDAM data set on a 3330-series drive attached to a block multiplexer channel. The records are fixed-length standard format and sector numbers are calculated from record ID (by data management).

Channel program 1. Initiate the seek and data transfer operation.

Command Chaining Flag	Command		Block Multiplexer Channel and Disk Control Unit Status
CC	SEEK	(Seek address)	Free during arm motion
CC	SET SECTOR	(Sector number of sector preceding desired record)	Free until sector found
CC	SEARCH ID EQ	(ID - sequential position on track)	Busy (260 microseconds average for a 3330)
CC	TIC	(Back to search if ID is not equal. With the logic shown, the first ID inspected normally is that of the desired record and the TIC command is not executed.)	Busy
	READ DATA	(Processor storage address of input area)	Busy

The preceding example indicates the advantages of rotational position sensing and block multiplexing:

- Only one channel program is required to locate a disk record and transfer the data, thereby eliminating a stand-alone seek I/O interruption and the I/O supervisor processing required to schedule a data transfer channel program. A channel-available interruption may occur, however, during channel program execution.

- The channel and disk control unit are free during arm motion and rotational positioning, allowing seek and set sector operations to be overlapped with other I/O operations on that control unit and channel. Implementation of multiple requesting permits a disk control unit to control concurrent execution of multiple RPS channel programs in order to overlap seek and set sector operations for its drives.

Performance improvement gains achieved on block multiplexer channels are not due entirely to the fact that direct access device rotational delays are overlapped. Also important is the ability to initiate seek commands a number of milliseconds earlier, because a block multiplexer channel is free. The initiation of stand-alone seeks on a selector channel is delayed during search and data transfer operations. On a block multiplexer channel, seeks can be initiated during rotational positioning, since the channel and disk control unit are not busy.

The concepts of rotational position sensing as described for the 3330-series also apply to the 2305 facility. Since an arm positioning seek is not required for the 2305, the channel program issued by an RPS access method or a user can be the same as that shown for the 3330-series except that the first command will not be a seek requiring mechanical motion.

SUMMARY OF BLOCK MULTIPLEXING OPERATIONS WITH I/O DEVICES

The following summarizes how direct access devices without and with RPS and other I/O devices operate on a block multiplexer channel on a Model 145 when executing a command-chained channel program.

1. Direct access devices without RPS (2311, 2321, 2314, 2303, and 2319) cannot be assigned to a UCW and operate in the same way whether the channel is in block multiplexer or selector mode. That is, the channel and the disk control unit are busy during the entire time a command-chained disk channel program is in operation. Thus, there is no disconnection after a chained seek.

2. A 3330-series string assigned to a nonshared UCW group on a block multiplexer channel and executing a command-chained channel program disconnects after the control unit accepts an arm-positioning seek that causes arm movement. Reconnection is attempted when the arm reaches its destination and signals device end. Disconnection also occurs when the control unit accepts a SET SECTOR command. When the sector specified arrives under the read/write heads, the control unit attempts to reconnect and resume the CCW chain. If the channel is busy, the control unit repeats the reconnection procedure each time the specified sector position is reached.

3. The 2305 facility assigned to a nonshared UCW group executing a command-chained channel program disconnects when the control unit accepts a SET SECTOR command. Reconnection occurs as described for the 3330-series.

4. All tape drives operate exactly the same whether the channel is in block multiplexer or selector mode, since they cannot be assigned a UCW. That is, the channel is busy during the entire time a command-chained channel program is in operation.

5. Buffered card and print devices (or devices operating with buffered control units) disconnect during the mechanical motion of the device. Reconnection occurs later to fill or empty the associated buffer. For example, a 1403 Printer attached to a 2821 control unit assigned to a nonshared UCW group on a channel operating in block-multiplexing mode disconnects from the channel during print time and carriage motion. Reconnection occurs when the channel is free to transfer the data for the next line to the 2821 buffer in burst mode. (These devices can also be assigned to a Type 3 group, in which case there will be no disconnection.)

6. Any other I/O device that presents channel end without simultaneous device end disconnects from a block multiplexer channel when command chaining if it has been assigned a nonshared UCW and its control unit is designed for such disconnection.

Section 60:15 discusses planning for installation of RPS devices on block multiplexer channels.

10:30 SYSTEM CONTROL PANEL AND SYSTEM CONSOLE

SYSTEM CONTROL PANEL

The system control panel attached to the end of the Model 145 CPU contains all the switches and indicators required to operate the system. The console layout--its location and the grouping of controls and indicators--was designed with human factors in mind. The five-foot-high CPU frame that contains the control panel permits most operators an unrestricted view of the machine area.

System controls are divided into three sections similar to the control sections of System/360 models. There is an operator control, an operator intervention, and a maintenance section.

A new item in the operator control section is the clock security lever switch, which is used in conjunction with programmed setting of the time of day clock. Other new buttons and switches have been added for control of the console file. They provide the capability of loading data (microcode or diagnostics) from the console file.

The capability of clearing processor storage from the system control panel is also provided. If the new enable system clear pushbutton is held in when the system reset or load button is pushed, processor storage, storage protect keys, and the general and floating-point registers are set to zero with correct ECC bits or parity.

SYSTEM CONSOLE

The microprogram-controlled 15-cps 3210 Model 1 Console Printer-Keyboard can be attached to the left-hand or right-hand extension of the Model 145 console reading board for use as the operator console device. Alternatively, the 3215 Model 1 Console Printer-Keyboard with a print speed of 85-cps can be used. The 1052 Model 7 Printer-Keyboard cannot be attached as the primary system console device.

These two printer-keyboards are functionally compatible and program compatible with each other and with the 1052. Their keyboards are the same as that of the 1052 except that the alternate coding key has been removed from the new printer-keyboards and the EOB (now called END) and cancel keys are separate pushbuttons.

Both the 3210 Model 1 and the 3215 Model 1 Console Printer-Keyboards have an alter/display mode of operation (not implemented for the 1052), which is of benefit to customer engineers and operators. After the system is placed in manual mode, this new mode is set by pressing the alter/display key, which places the console under microprogram control. In this mode, data can be placed in, or printed from, the following:

- Processor storage
- General, floating-point, and control registers
- The current PSW
- A storage protect key
- Local or control storage (display only unless the CE key switch is on)

A mnemonic is entered to indicate the function to be performed. Other data, such as the starting storage address and the data to be entered, must be supplied in hexadecimal format. Both uppercase and lowercase may be used. If an error is made (such as incorrect mnemonic, storage address, or hexadecimal data characters), the microprogram detects the error and the operation can then be restarted. The microprogram also handles carriage returns automatically during data displays and prints eight words of hexadecimal characters per line until the end key is depressed.

The 3210 Model 2 Adapter optional feature allows a 3210 Model 2 Console Printer-Keyboard to be connected to the system to function as a remote alternate or additional console (either a 3210 Model 1 or a 3215 Model 1 can be the primary console). The 3210 Model 2 can be located up to 75 feet from the CPU. This remote console cannot be used for alter/display operations.

Store status is another function that can be performed using the console typewriter keyboard in alter/display mode. The operator can cause certain status information to be placed in processor storage by entering the console typewriter mnemonic ST. The contents of the following are placed in processor storage during a store status operation:

CPU timer - locations 216-223
Clock comparator - locations 224-231
Current PSW - locations 256-263
Floating-point registers - locations 352-383
General registers - locations 384-447
Control registers - locations 448-511

The operator should perform the store status function to perserve system status after an error causes a system halt and prior to resetting the system to load a stand-alone 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 stand-alone dump programs provided will be modified to obtain system status information from the locations indicated.

10:35 STANDARD AND OPTIONAL SYSTEM FEATURES

STANDARD FEATURES

Standard features for the System/370 Model 145 are:

- Instruction set that includes binary and decimal arithmetic instructions, the new general purpose instructions, and the instructions required to handle new standard features. New standard instructions for the System/370 Model 145 (not available on Models 30 and 40) are:

  COMPARE LOGICAL CHARACTERS UNDER MASK
  COMPARE LOGICAL LONG
  *HALT DEVICE
  INSERT CHARACTERS UNDER MASK
  *LOAD CONTROL
  *LOAD REAL ADDRESS
  MONITOR CALL
  MOVE LONG
  *PURGE TLB
  *RESET REFERENCE BIT
  *SET CLOCK
  SHIFT AND ROUND DECIMAL
  *STORE CHANNEL ID
  STORE CHARACTERS UNDER MASK
  STORE CLOCK
  *STORE CPU ID
  *STORE CONTROL

*Privileged instructions

*STORE THEN AND SYSTEM MASK
*STORE THEN OR SYSTEM MASK
(START I/O FAST RELEASE, a System/370 privileged instruction functionally implemented on the Models 165 and 168, is executed as a START I/O on the Model 145)

- BC and EC modes of operation
- Dynamic address translation
- Channel indirect data addressing
- Microinstruction retry
- Monitoring feature
- Program event recording
- Interval timer (3.3 millisecond resolution)
- Time of day clock
- Expanded machine check interruption class
- Interruption for SSM instruction
- Reloadable control storage - 32K
- Movable control storage boundary
- ECC on processor and control storage
- Byte-oriented operands
- Store and fetch protection
- Byte multiplexer channel (16 subchannels assumed but not contained in basic system microcode)
- Selector channel 1 (IFA not present) or selector channel 2 (IFA installed)
- Channel retry data in a limited channel logout area
- OS/DOS Compatibility feature
- Console file for microcode and microdiagnostic routine loading

*Privileged instructions

OPTIONAL FEATURES

Optional features for the System/370 Model 145, which can be field installed unless otherwise noted, are:

- CPU timer and clock comparator (and associated privileged instructions SET CPU TIMER, STORE CPU TIMER, SET CLOCK COMPARATOR, STORE CLOCK COMPARATOR)
- Floating-point arithmetic including extended precision (no-charge feature)
- 1401/1440/1460 Compatibility (no-charge feature)
- 1401/40/60, 1410/7010 Compatibility (no-charge feature)
- Direct Control (includes External Interrupt feature)
- Byte multiplexer subchannels above 16 (no-charge feature)
- Integrated File Adapter with 2319 Disk Storage - channel 1 position only (not recommended for field installation but may be removed in the field)
- Selector channel 3 (in addition to IFA)
- Selector channels 2, 3, and 4 (if IFA is not installed)
- Channel Word Buffer feature on all installed selector channels
- Block multiplexer mode for any or all installed selector channels (no-charge feature)
- Channel-to-Channel Adapter
- 3210 Model 1 Adapter** - for attachment of a 3210 Model 1 console
- 3210 Model 2 Adapter - for attachment of a remote 3210 Model 2 console
- 3215 Model 1 Adapter** - for attachment of a 3215 Model 1 console
- 3345 Main Storage Frame Model 1 or 2***
- 3345 Storage and Control Frame Model 3, 4, or 5***
- 3046 Power Unit - required for processor storage configurations of 384K and 512K

**Either the 3210 Model 1 or the 3215 Model 1 Console Printer-Keyboard must be installed as the primary system console.
***Maximum one 3345 per system.

SECTION 15: VIRTUAL STORAGE AND DYNAMIC ADDRESS TRANSLATION

The first subsection, 15:05, 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 15:05 defines the terminology associated with virtual storage and dynamic address translation hardware. The terminology included is that common to the IBM-supplied operating systems that support a virtual storage environment for System/370. Terms unique to a particular operating system are defined in the optional supplement that describes that operating system.

Subsection 15:10 describes in detail the implementation and operation of dynamic address translation and channel indirect data addressing in the Model 145. Other hardware items associated with dynamic address translation, such as reference and change recording, are discussed also.

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

The optional programming systems supplements (Sections 80 to 110) assume knowledge of the entire contents of Section 15.

15:05 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, 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 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 in 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 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, therefore, 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 15.05.1). 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.

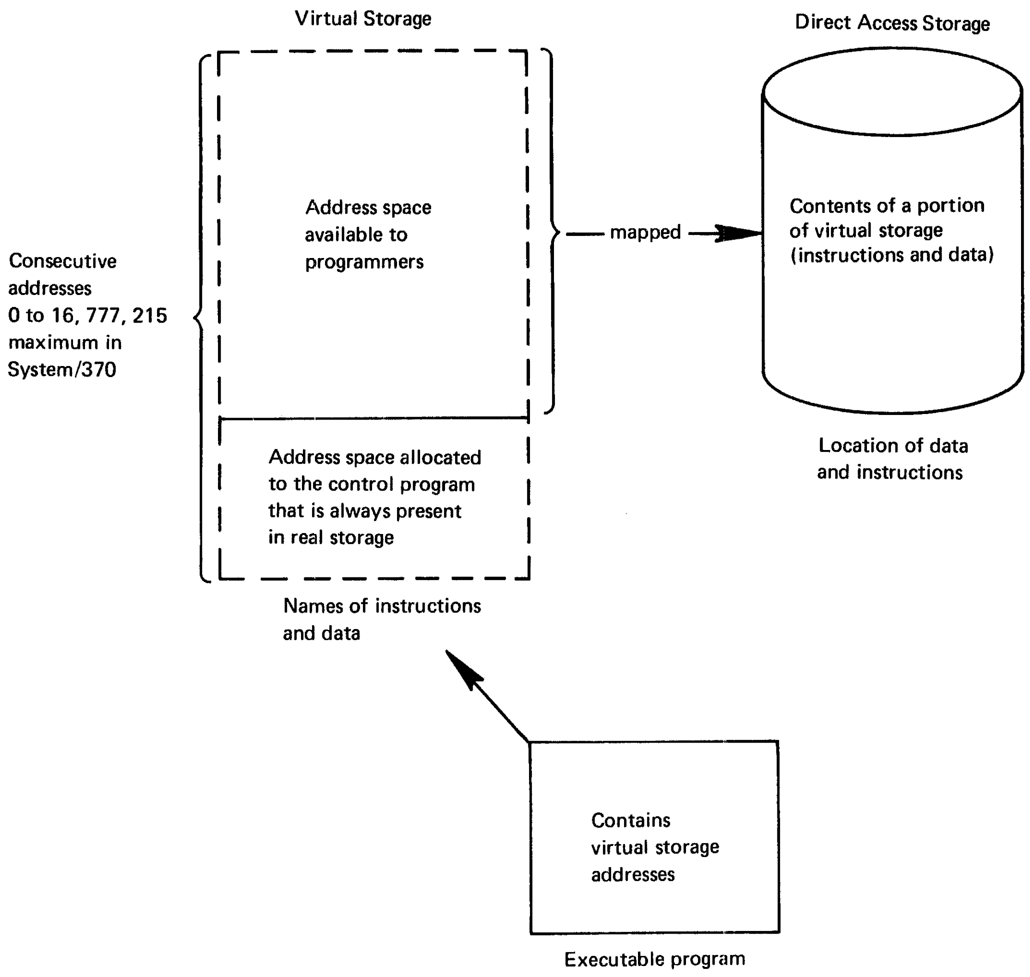

Virtual Storage
Direct Access Storage
Address space available to programmers
mapped
Contents of a portion of virtual storage (instructions and data)
Consecutive addresses 0 to 16, 777, 215 maximum in System/370
Address space allocated to the control program that is always present in real storage
Location of data and instructions
Names of instructions and data
Contains virtual storage addresses
Executable program

Figure 15.05.1. 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, direct access storage 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 a 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 back 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 15.05.2 illustrates the relationship of virtual storage, direct access storage, and real storage without regard for 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.

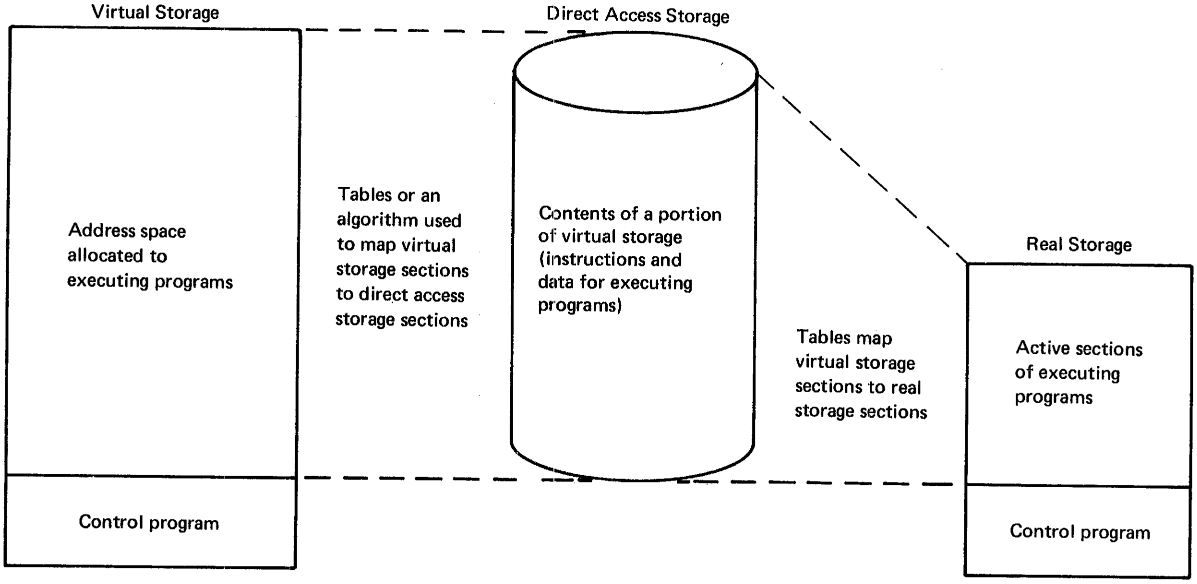

Virtual Storage
Address space allocated to executing programs
Control program
Tables or an algorithm used to map virtual storage sections to direct access storage sections
Direct Access Storage
Contents of a portion of virtual storage (instructions and data for executing programs)
Tables map virtual storage sections to real storage sections
Real Storage
Active sections of executing programs
Control program

Figure 15.05.2. 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 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 15:15.

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, if any, index values 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 programmed translation is required prior to 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 by the program fetch routine, 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, OS MFT, and OS 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 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 that the technique of using direct access storage and DAT hardware to support a larger address space has over using larger real storage.

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 only one system. Multiple virtual storages can be used to support multiple virtual machines. A discussion of virtual machines is contained in Virtual Machine Facility/370 (VM/370): Introduction (GC20-1800).

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 such that the need for transferring program sections into and out of real storage is kept at an acceptable level.

As previously mentioned, most programs can be structured such 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 frequently referenced 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 fact 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 to be executed. For example, a given 150K language translator might be able to operate with an average of 75K of real storage dynamically available to it during its operation; however, the time required to compile a program on an intermediate-scale model under these conditions might be unacceptable. Alternatively, the performance desired on the intermediate-scale CPU might be achieved if an average of 100K is dynamically available to the language translator while it operates. Without a virtual storage operating system, the 150K language translator probably could not be used at all on the intermediate-scale model because of its relatively large design point size.

The availability of lower cost real storage for the Model 145 and the real storage independence that a virtual storage environment offers provide new flexibility in tradeoffs among real storage cost, function, and individual 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 too, 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 skill and knowledge. 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 System/370 models with relatively 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 smaller scale 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 can 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 job in highest real storage, for example.

  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 that can occur concurrently, 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 more 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 15.05.3.

In existing mixed batch and online-oriented installations, dynamic real storage management allows programming techniqes 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 the entire time the applications are active.

Figure 15.05.3 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.

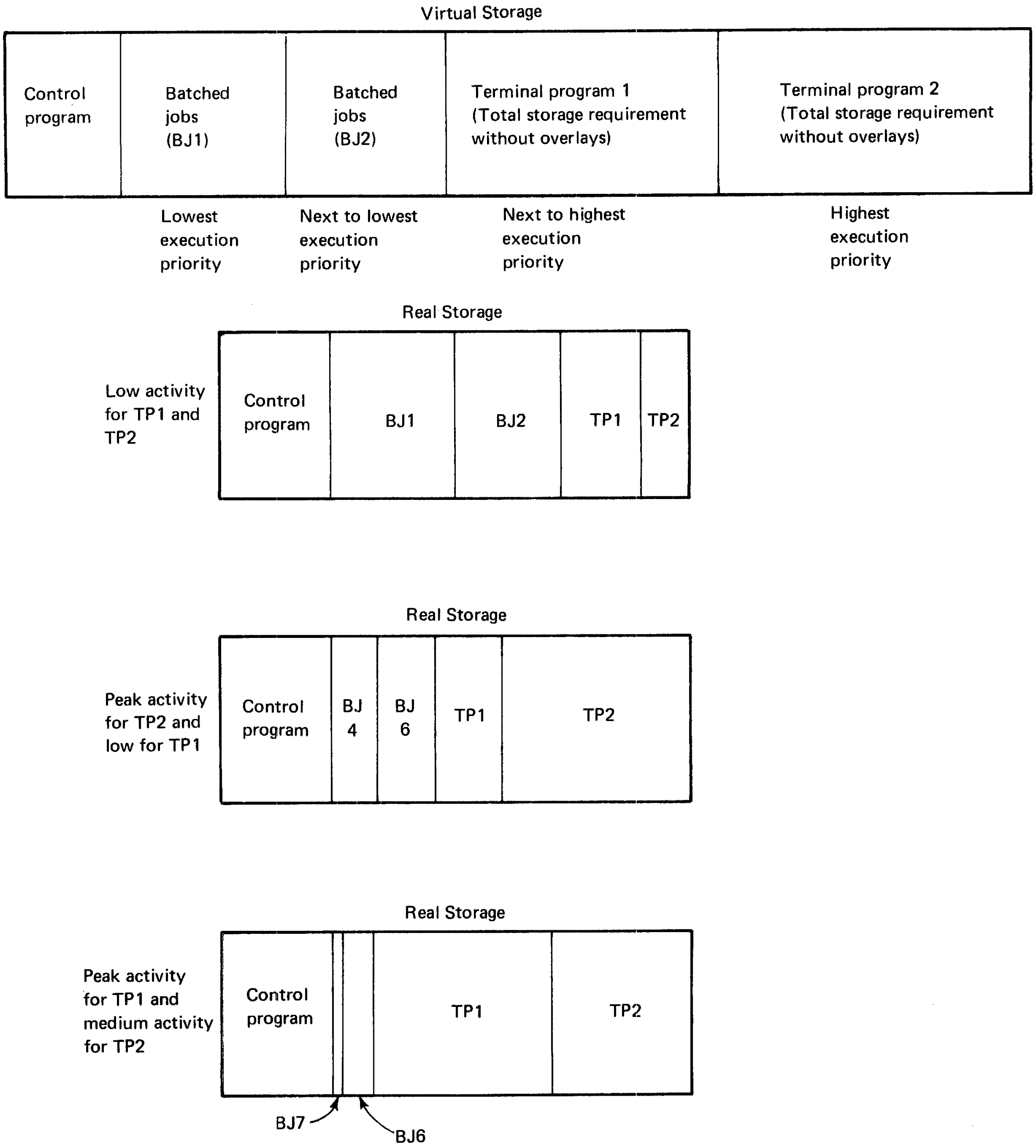

Virtual Storage
Control program
Batched jobs (BJ1)
Batched jobs (BJ2)
Terminal program 1 (Total storage requirement without overlays)
Terminal program 2 (Total storage requirement without overlays)
Lowest execution priority
Next to lowest execution priority
Next to highest execution priority
Highest execution priority
Real Storage
Low activity for TP1 and TP2
Control program
BJ1
BJ2
TP1
TP2
Real Storage
Peak activity for TP2 and low for TP1
Control program
BJ 4
BJ 6
TP1
TP2
Real Storage
Peak activity for TP1 and medium activity for TP2
Control program
TP1
TP2
BJ7
BJ6

Figure 15.05.3. **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 the 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 15:15). 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 thereby 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 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 System/370 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 that 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 also is 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 15.05.4 illustrates the relationship of virtual storage, external page storage, and real storage that was conceptually shown in Figure 15.05.2. (Note that the terms swap-in, swap-out, and working set have a specific meaning in a time-sharing environment and are defined in OS/Virtual Storage 2 Features Supplement under "Time Sharing Option".)

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.

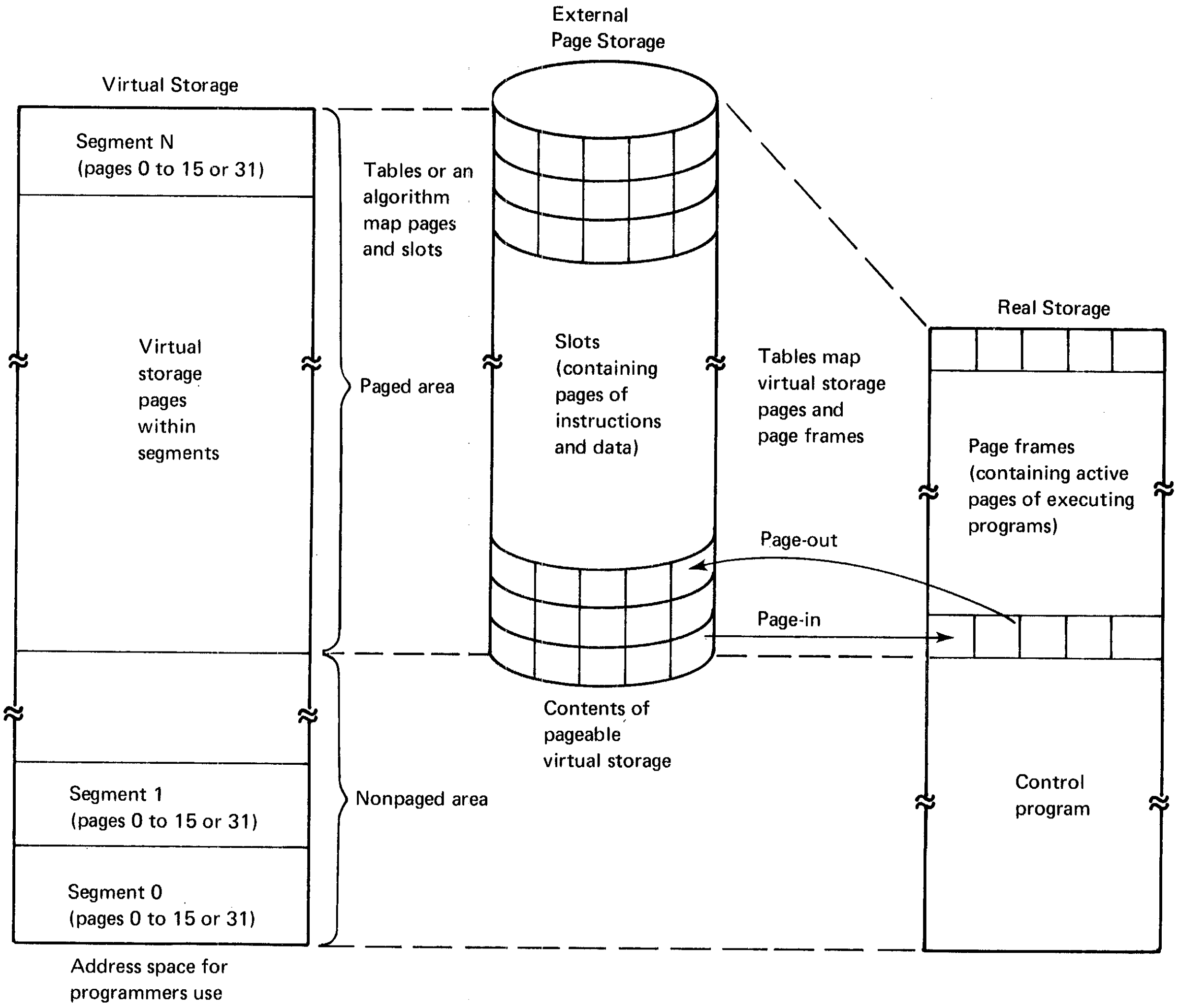

External Page Storage
Virtual Storage
Segment N (pages 0 to 15 or 31)
Tables or an algorithm map pages and slots
Virtual storage pages within segments
Paged area
Slots (containing pages of instructions and data)
Tables map virtual storage pages and page frames
Real Storage
Page frames (containing active pages of executing programs)
Page-out
Page-in
Contents of pageable virtual storage
Segment 1 (pages 0 to 15 or 31)
Nonpaged area
Control program
Segment 0 (pages 0 to 15 or 31)
Address space for programmers use

Figure 15.05.4. Layout of virtual storage, external page storage, and real storage

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.

While page-ins usually are initiated as a result of a page fault, OS/VS1 and OS/VS2 provide an Assembler Language 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 incurring any page faults. Use of this macro is restricted because unlimited use of this facility can defeat the objective of demand paging. DOS/VS does not support a page-ahead facility.

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 implements 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.

Prior to 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. One reason is for better operation of the system. This reason applies to all (in DOS/VS) or a portion (in OS/VS1 and OS/VS2) of the control program, certain 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, in order for the operation to 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 operating systems support both long-term fixing and short-term fixing. 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 prior to 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 indicated previously, 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 are not placed in external page storage (because they are not paged) even though they are allocated space in virtual storage. In OS/VS2 and OS/VS1, 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. 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 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 15:05.

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 15:10 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 executing 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 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 loaded 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 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 dynamically loaded 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.

15:10 DYNAMIC ADDRESS TRANSLATION FACILITY FOR THE MODEL 145

Dynamic address translation is a standard facility of the Model 145. DAT hardware 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 145, and programmed channel program translation are discussed later in this subsection under "Channel Indirect Data Addressing".

The following instructions, all privileged, are associated with dynamic address translation: LOAD REAL ADDRESS, RESET REFERENCE BIT, and PURGE TLB. These instructions are valid when either BC or EC mode is in effect. They operate identically in both modes.

VIRTUAL STORAGE ORGANIZATION

The Model 145 (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 15.10.1 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 specifiy 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 or 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.

Table 15.10.1. Number and size of segments and pages for a 16-million-byte virtual storage

CR 0 Bits 11,12	 8,9	Segment Size in Bytes	Number of Segments in the Virtual Storage	Page Size in 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

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 15.10.1.

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 or not 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 translation exception occurs during the translation process.

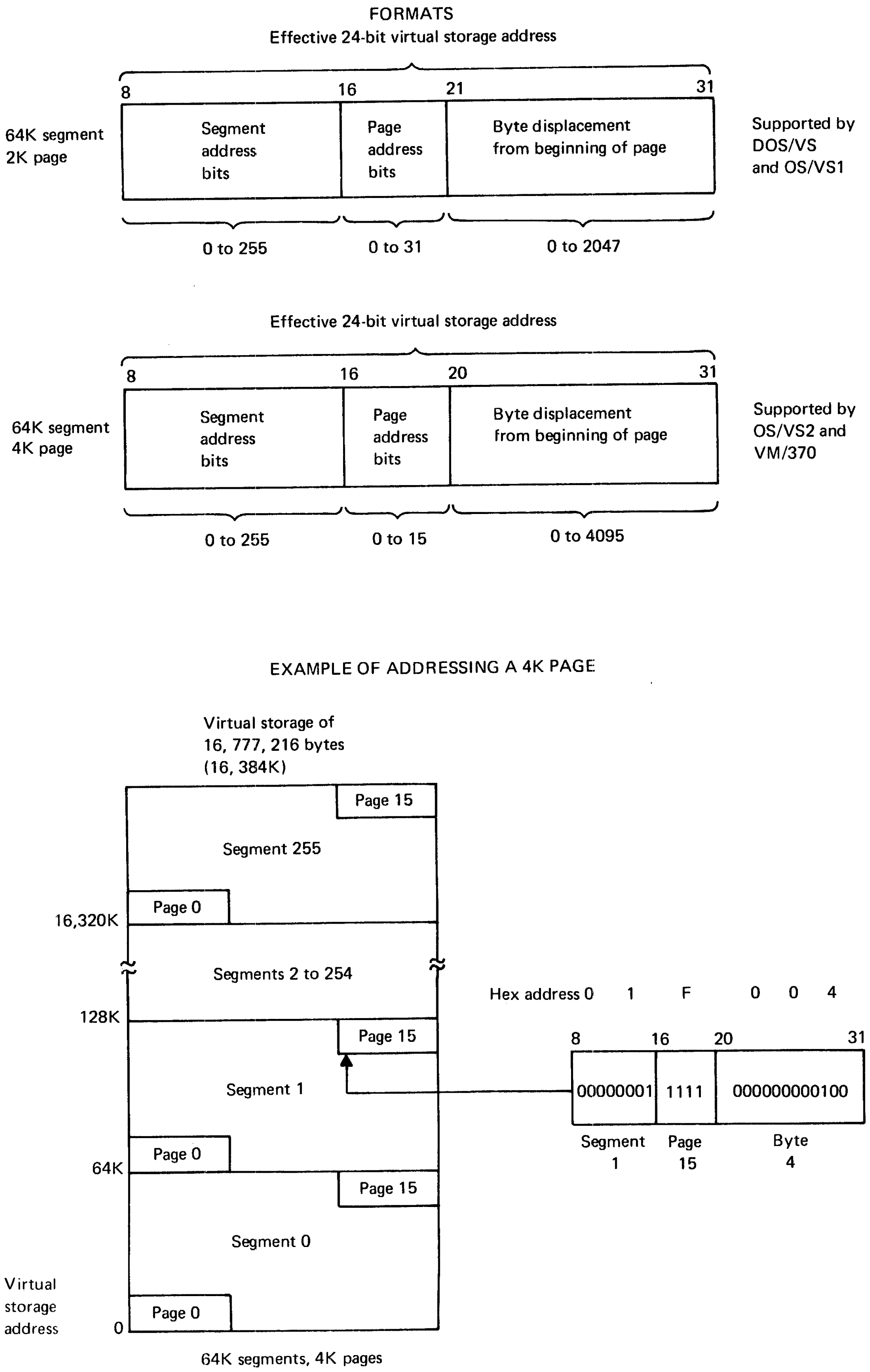


Figure 15.10.1. Virtual storage address fields for a 64K segment

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 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 15.10.2. 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, which must be a multiple of the segment size (64K for IBM-supplied support), 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 storage is dynamically allocated and deallocated, the segment table constructed during IPL is modified as required 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 and data addresses contained in other instructions. Implicit address translation takes place during instruction execution.

The details of the translation process are given in Figure 15.10.3. 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 after an explicit translation request (LOAD REAL ADDRESS instruction) are indicated via the condition code setting instead of via an interruption.

Translation Lookaside Buffer

When the Model 145 operates with DAT specified, additional CPU time is required to perform the address translation process. In the Model 145, a translation lookaside buffer (TLB) is implemented to eliminate address translation time when possible.

The TLB contains a set of eight associative registers. Each register has two fields. The segment and page field can contain the high-order address bits of a recently translated virtual storage address. The real address field contains the 12 or 13 high-order address bits of the page frame currently assigned to the virtual storage page. Every time a virtual storage address is translated during instruction execution, the virtual storage address and the resulting real storage address are placed in the TLB. A least-recently-used algorithm is implemented to determine the register in which the new translation is placed.

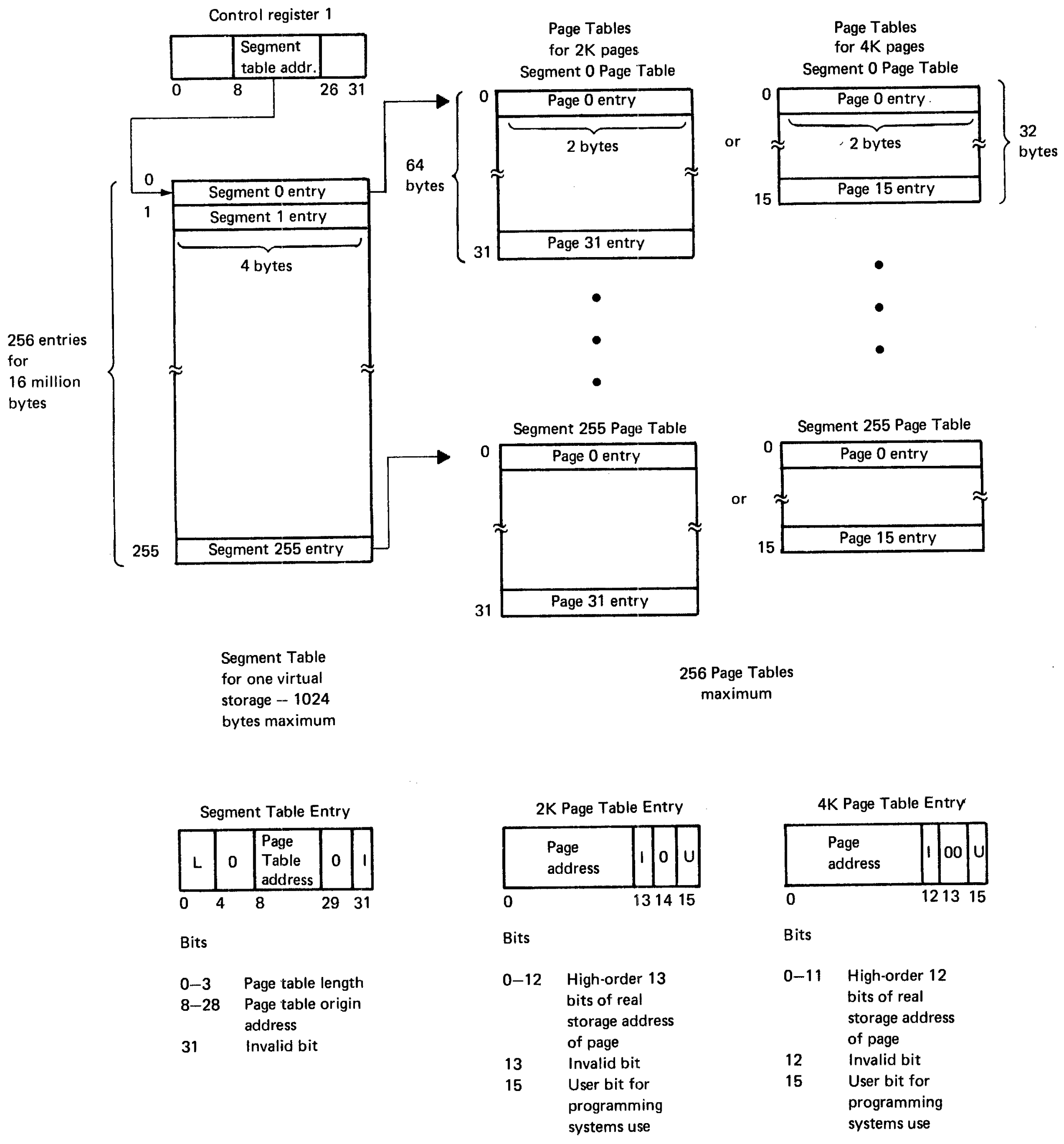
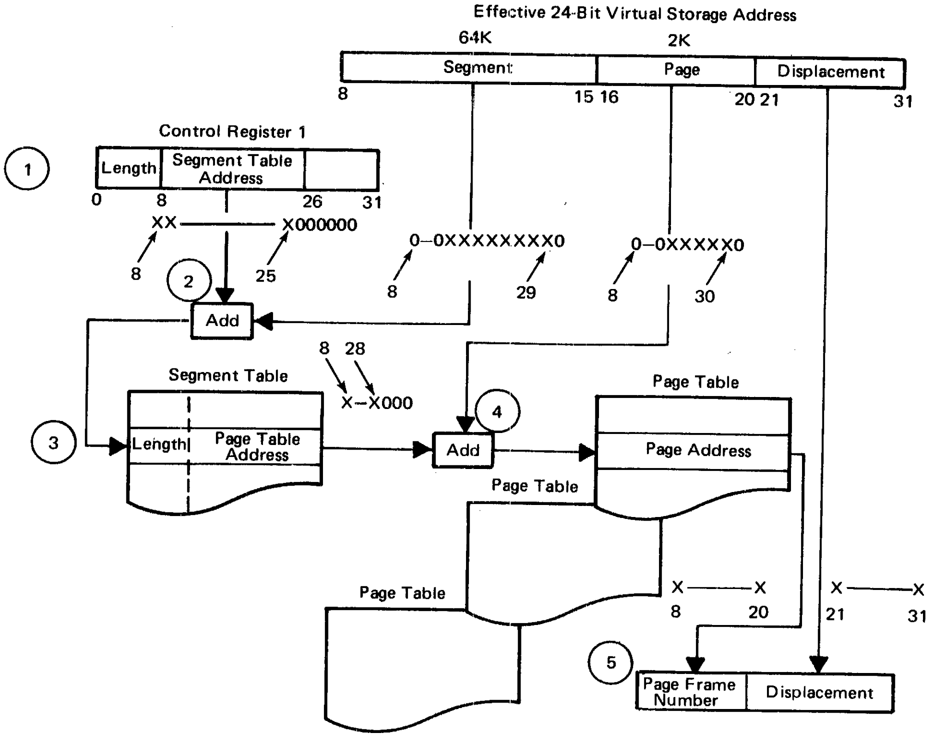


Figure 15.10.2. Segment table and page tables used for dynamic address translation



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 address 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 bits from the virtual storage address.

Figure 15.10.3. Dynamic address translation procedure

After the effective virtual storage address has been computed and prior to performing translation using the tables, the TLB is interrogated to determine whether or not it contains the translated address. The eight registers are inspected simultaneously. The segment and page bits from the virtual storage address to be translated are compared to the eight segment and page fields in the TLB registers. If a match occurs, the real storage address is taken from the TLB register and translation processing is finished. No additional CPU time is required for the translation process when the real address is obtained from the TLB. If the TLB does not contain the required translation, the complete translation procedure, as previously described, is performed, which requires four microseconds.

During initial program reset, the virtual storage address and the parity bit in each of the eight registers in the TLB are set to zero. Thus, no match can occur until new values are loaded as a result of the translation process. The PURGE TLB instruction provides the capability of clearing all eight TLB registers during system operation. In general, this instruction must be issued when an entry in a page table is invalidated, since the real storage address bits being invalidated could be contained in the TLB. (The control program will purge the TLB as required.) The TLB is also cleared when the DAT mode bit is turned off, when control register 1 is changed, when segment or page size is changed, and when certain mode changes are made by the DOS emulator program.

Since the TLB can hold eight translation addresses at any time, a task can work with a given 16K or 32K of program and data during an interval when it has CPU control without invoking the table lookup translation procedure after the eight required virtual storage addresses have been translated.

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 that 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 15.10.2 lists, for the Model 145, those storage addresses designated by a program, either explicitly or implicitly, that are virtual (and, therefore, are subject to translation) and those storage 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 thus, which are real.

Table 15.10.2. Virtual and real storage addresses used by and supplied to programs in the Model 145

<u>Virtual Storage Addresses Explicitly Designated by the Program</u> (translated)

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

<u>Real Storage Addresses Explicitly Designated by the Program</u> (not translated)

- Operand addresses in SET STORAGE KEY, INSERT STORAGE KEY, and RESET REFERENCE BIT instructions
- Machine check extended log (MCEL) pointer in control register 15
- I/O extended log (IOEL) pointer in location 172
- 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

<u>Addresses Not Used to Address Storage</u> (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)

<u>Real Storage Addresses Used Implicitly</u> (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

<u>Virtual Storage Addresses Provided to the Program</u>

- 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

<u>Real Storage Addresses Provided to the Program</u>

- The translated address generated by the LOAD REAL ADDRESS instruction
- Address of a 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 Console Printer-Keyboard

The alter/display function of the console printer-keyboard is expanded in the Model 145 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.

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

The STORE and DISPLAY console panel functions still use real storage addresses only. The ADDRESS COMPARE and SET IC functions assume the storage address indicated to be virtual or real, depending on the setting of the select switch.

FEATURES TO SUPPORT DEMAND PAGING

Reference and Change Recording Facility for Real Storage Blocks

A hardware recording facility is standard in the Model 145. 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 145 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 by either 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 console printer-keyboard and store/display operations initiated from the console panel 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.

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. In order for the instruction to operate correctly the second time, execution of the instruction must have been stopped such that reexecution gives the same results as would occur 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 such that no operation is performed, no fields are 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, execution of the instruction 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 145 by additional microcode routines that are executed prior to normal instruction execution microcode.

Instructions that do not need pretesting 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 the 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 three). 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 that 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 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 prior to the initiation of I/O operations. 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. It applies to the byte multiplexer channel, selector channels, block multiplexer channels, and the IFA. As shown in Figure 15.10.4, the use of channel indirect data addressing allows the channel program logic used in the CCW list that contains 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 backwards, 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 real 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 15.10.4 shows an example of the IDAL's required for a command-chained CCW list. Page size used is 2K. 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.

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.

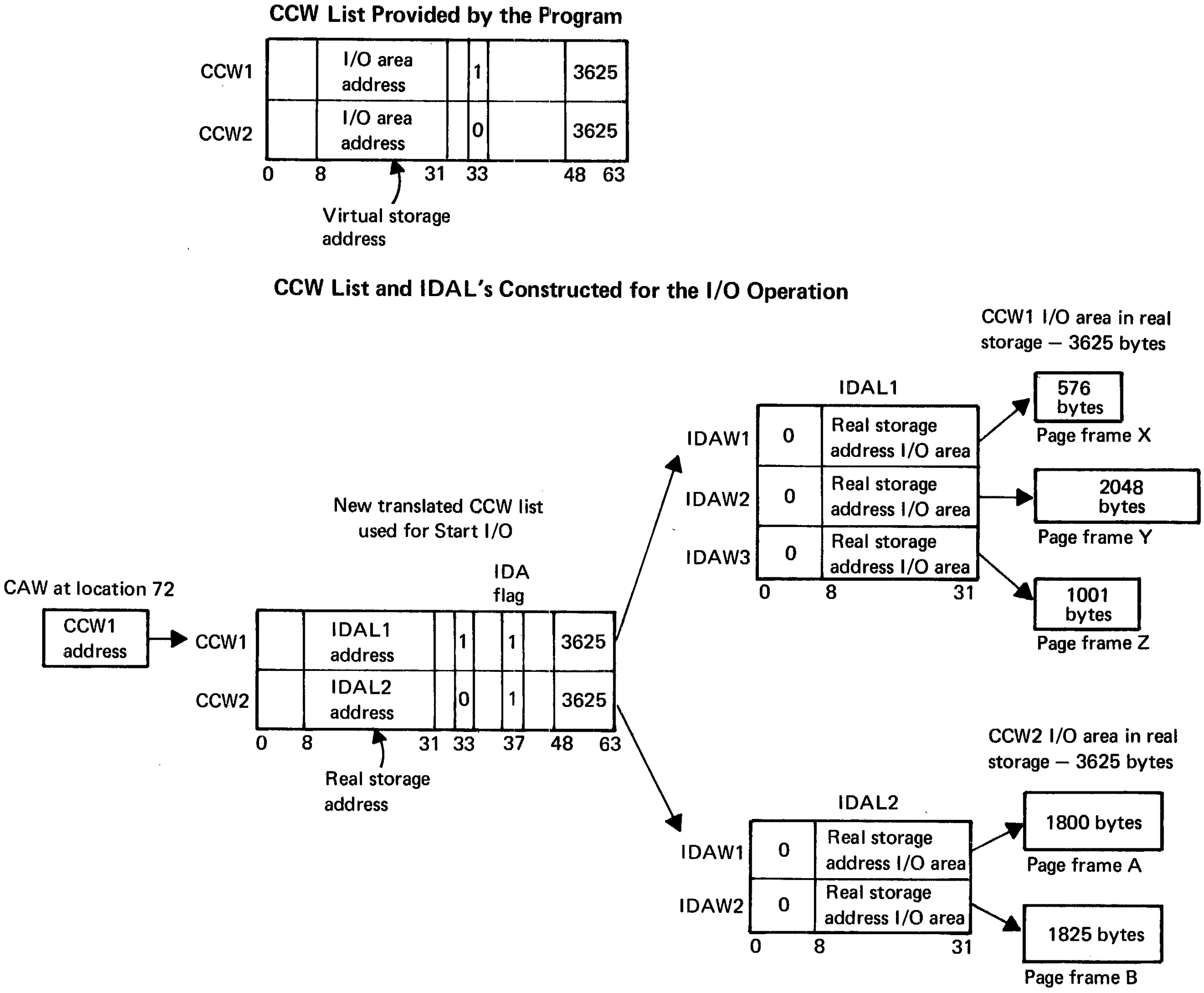

CCW List Provided by the Program
CCW1
I/O area address
1
3625
CCW2
I/O area address
0
3625
0
8
31
33
48
63
Virtual storage address
CCW List and IDAL's Constructed for the I/O Operation
CCW1 I/O area in real storage — 3625 bytes
IDAL1
IDAW1
0
Real storage address I/O area
IDAW2
0
Real storage address I/O area
IDAW3
0
Real storage address I/O area
0
8
31
576 bytes
Page frame X
2048 bytes
Page frame Y
1001 bytes
Page frame Z
New translated CCW list used for Start I/O
IDA flag
CAW at location 72
CCW1 address
CCW1
IDAL1 address
1
1
3625
CCW2
IDAL2 address
0
1
3625
0
8
31
33
37
48
63
Real storage address
CCW2 I/O area in real storage — 3625 bytes
IDAL2
IDAW1
0
Real storage address I/O area
IDAW2
0
Real storage address I/O area
0
8
31
1800 bytes
Page frame A
1825 bytes
Page frame B

Figure 15.10.4. Example of IDAL's required for a CCW list when page size is 2K

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 145, IDAW's are prefetched.) The channel continues using the IDAL until the operation indicated by the CCW completes (CCW count reaches zero, IRG on tape is 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 addresses must be built, and the appropriate IDAL's must be constructed prior to 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 assigned. If not, ensure that page frames are allocated and fixed.

At the completion of the I/O operation, the real storage address in the channel status word must be translated to a virtual storage address, and the pages that were short-term fixed prior to the 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 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.

15:15 SYSTEM PERFORMANCE IN A VIRTUAL STORAGE ENVIRONMENT

A virtual storage environment is designed to provide new data processing capabilities. As is true about 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 145, 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 15:10. 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, 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 prior to 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 or not 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, for example.

- 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.

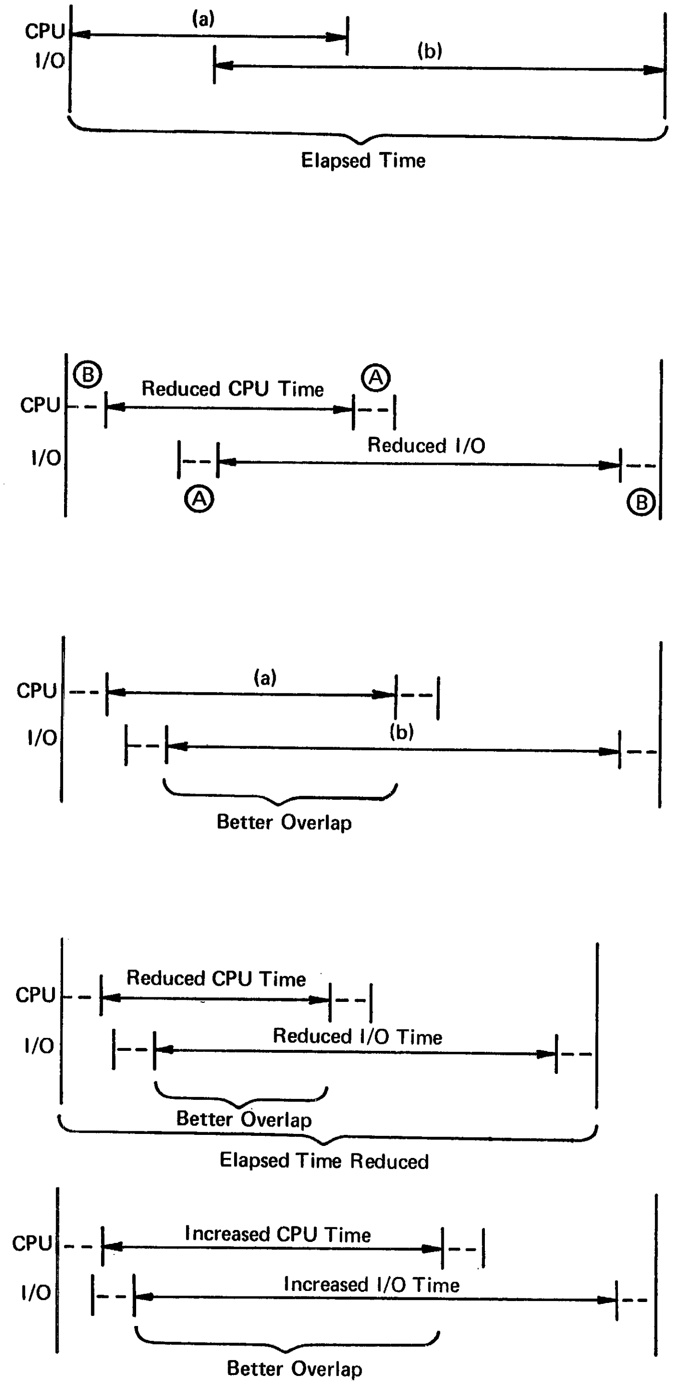
- Direct access storage is required for external page storage. The amount required depends on the amount of virtual storage that is to be supported and the way in which the particular operating system organizes and manages external page storage. (See the programming systems supplements for external page 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 significance of this increase will be greater in systems with smaller real storage sizes.

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 unoverlapped, 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 the operating system provides as well as by an increase in resource utilization that is required to support a virtual storage environment.

Figure 15.15.1 conceptually illustrates possible system performance when a virtual storage operating system is installed on an existing Model 145 configuration. A sample throughput obtained using a nonvirtual storage operating system is shown in panel 1. (It is not meant to represent any specific Model 145 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.



Panel 1

Sample existing CPU and I/O utilization and overlap for a Model 145.

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 15.15.1. Possible system performance when a virtual storage operating system is used with an existing Model 145 configuration

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 using 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 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 system 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 given, 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 is 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.

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, such 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 (partition or region) in deactivated status. When the page frames associated with a deactivated partition or region become available, they can be allocated to other tasks to reduce paging activity. Later, when paging activity becomes sufficiently low, the deactivated partition or region 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 145 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 145 can execute many more instructions during a page-in from a 2314, for example, than can a Model 135. 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 the 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 15.15.2, 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 occurs 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 requires more real storage when a relatively slower paging device is used than if a faster paging device is used.

Program Structure. The total amount of virtual storage a program uses is not nearly as 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 such 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 naturally have a 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 15.15.2 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 15.15.2 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 15.15.3 illustrates how the paging factor only generally affects system performance. Figure 15.15.5, 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 15.15.2 and 15.15.3 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.

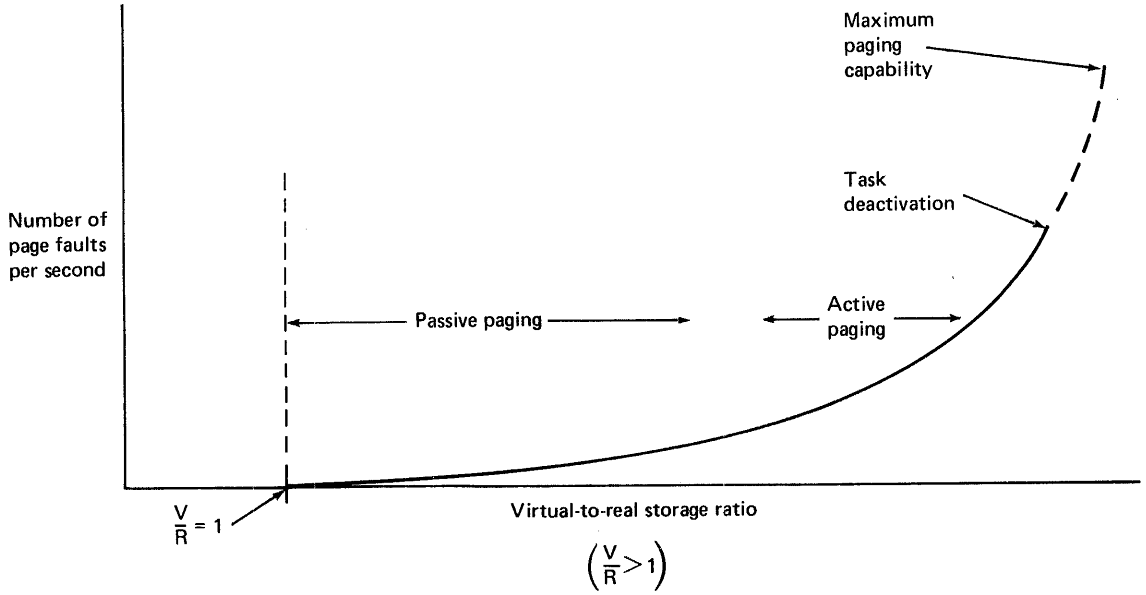


Figure 15.15.2. General effect on page faults of increasing the ratio of the virtual storage used to the real storage present in the system.

Figure 15.15.4 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.

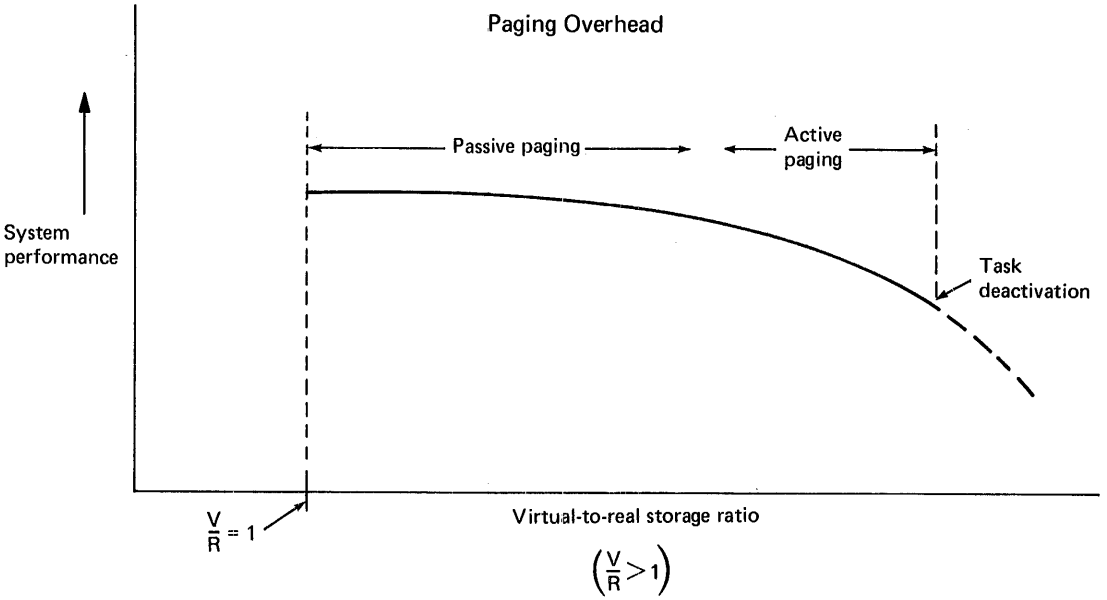


Figure 15.15.3. General effect on system performance of the paging factor only

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 such that real storage requirements and page faults are minimized. 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 and/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.

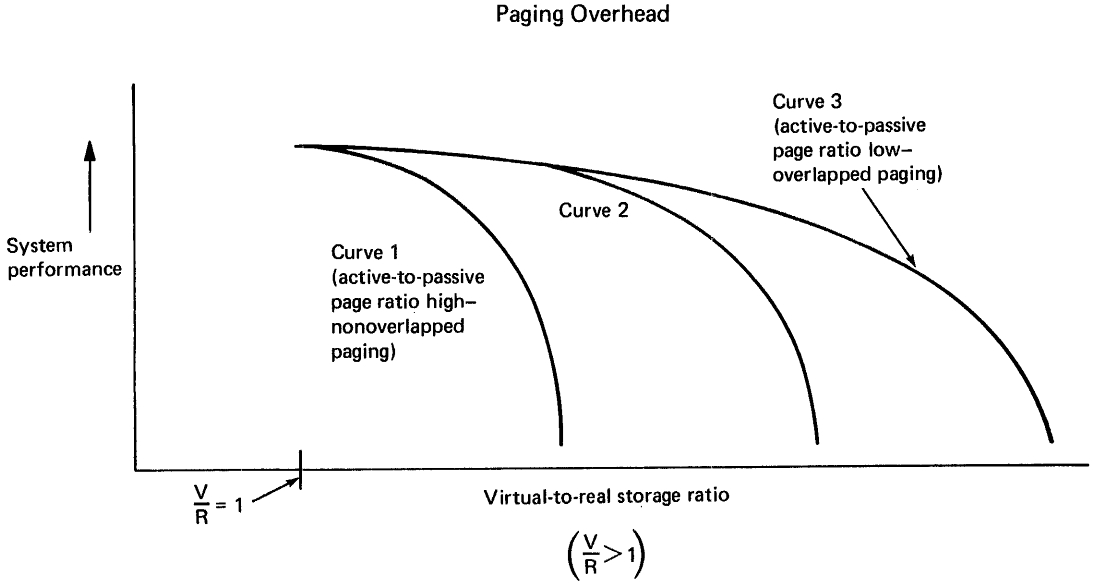


Figure 15.15.4. General effect of the paging factor on system performance for various active-to-passive page ratios.

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 15.15.5. 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.

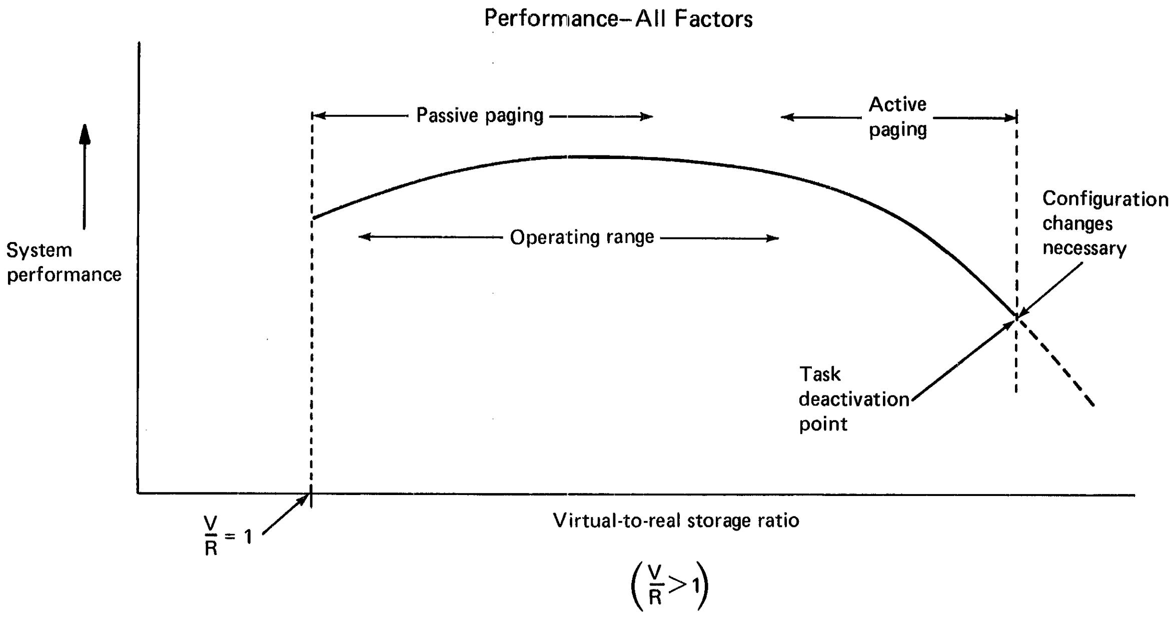


Figure 15.15.5. General system performance curve for a virtual storage environment

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:

- Install the optional Channel Word Buffer feature on the Model 145 to reduce channel interference with the CPU and, thereby, reduce existing CPU utilization.

- 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 prevented the use of larger buffer sizes in general and 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 considerable 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 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 makes it more suited to a virtual storage environment than the code produced by 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 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 and DOS 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 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, use of POWER in DOS), (2) operating more regions or partitions concurrently, or (3) increasing the use of multitasking (structuring a DOS/VS QTAM or 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 and 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 a 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 is 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 20: I/O DEVICES

20:05 I/O DEVICE SUPPORT

Most presently announced I/O devices and telecommunications terminals that can be attached to System/360 Models 25, 30, and 40 can be attached to the System/370 Model 145. The following I/O devices and features are not included in standard Model 145 configurations:

1052-7	Printer-Keyboard
1231	Optical Mark Page Reader
1285	Optical Reader
1404	Printer
1412	Magnetic Character Reader
1418	Optical Character Reader
1428	Alphameric Optical Reader
1445	Printer
1827	Data Control Unit (for attachment of 1800 system analog and/or digital control units to the Model 145)
2150	Console
2301	Drum Storage
2302	Disk Storage
2319-A3	Disk Storage
7340	Hypertape Drive
7772	Audio Response Unit

Selective Tape List feature on the 1403 Printer

The 1287 Optical Reader and 1288 Optical Page Reader can be attached to a byte multiplexer channel only. In addition, 2361 Core Storage cannot be attached to a Model 145.

I/O devices for the Model 145 announced with System/370 models are:

- 3330-series disk storage - attaches to a selector or a block multiplexer channel on the Model 145
- The 2305 facility Model 2 - attaches to a block multiplexer channel on the Model 145
- The 3211 Printer - attaches to any Model 145 channel
- The 3803/3420 and 3410/3411 Magnetic Tape Subsystems - attach to any Model 145 channel
- The 3505 Card Reader and the 3525 Card Punch - attach to any Model 145 channel

The 3330-series and the 2305 represent significant advances in direct access device technology. They provide larger online data capacity, faster data rates and access, and expanded error correction features. Both have rotational position sensing and multiple requesting as standard features.

The 3211 Printer offers almost twice the print speed of the 1403-N1 and new features designed to reduce operator intervention.

The 3410/3411 and 3803/3420 tape subsystems incorporate advances in tape speed, design, and technology and offer new features, new design, and enhanced reliability, availability, and serviceability to 2400-series magnetic tape unit users.

The 3505 and 3525 80-column card units, embodying many new design features to enhance reliability, offer a variety of speeds and many new operator-oriented facilities.

The major new characteristics of these devices are discussed in the following subsections.

20:10 3330-SERIES DISK STORAGE

3330 DISK STORAGE AND 3333 DISK STORAGE AND CONTROL

The 3330-series is large-capacity, high-performance direct access storage that consists of 3330 Disk Storage and 3333 Disk Storage and Control. A 3330-series string can consist of one to four modules (from two to eight drives), and is connected to a Model 145 selector or block multiplexer channel via 3830 Storage Control (Model 1 or 2) or via integrated storage control contained in a storage frame connected to the Model 145 CPU. The removable 3336 Disk Pack is used for data storage. Track overflow, rotational position sensing, and multiple requesting are standard features of 3830 Storage Control and integrated storage control. The block multiplexing feature must be installed on the Model 145 in order to use multiple requesting and RPS.

A 3330-series string can consist of 3333 Disk Storage and Control and/or 3330 Disk Storage modules. A 3333 Disk Storage and Control module consists of two independent drives and device-oriented control functions. It must be the first unit in a 3330-series string attached to a Model 145 channel via integrated storage control (ISC) or via 3830 Storage Control Model 2. The 3333 does not attach to 3830 Storage Control Model 1. The 3330 Disk Storage module consists of two drives and does not contain the device-oriented control functions that are part of a 3333 module. A 3330 module connects directly to 3830 Storage Control Model 1 and to 3333 modules. Up to three 3330 modules can be attached to a 3333 module. Functionally, all drives operate in the same way, regardless of whether they are part of 3333 or a 3330 module.

The 3336 Model 1 Disk Pack is used with 3330 and 3333 modules. It has 19 recording surfaces and a maximum capacity of 100 megabytes when full-track records are used. A 3330-series string can consist of two, four, six, or eight drives for a maximum of 800 million bytes per string. The removable 3336 Model 1 Disk Packs are interchangeable across all 3330 and 3333 modules. Model 1 3336 Disk Packs are not interchangeable with the 2316 Disk Packs used on 2314 disk drives. (Table 20.10.2 compares disk pack characteristics.)

Each drive in a 3330 or 3333 module is mounted in a powered drawer that is opened and closed by a switch on the operator's panel contained in each module. Each panel provides the switches and indicators associated with individual drives in the module. Included on each operator panel are write inhibit switches, one for each drive. A switch can be individually set to permit both reading and writing or reading only on its associated drive. When an attempt is made to write on a drive that is set for read operations only, an interruption occurs and IBM-supplied programming support terminates the program that attempted to write on the protected drive. Each operator panel also contains a removable logical address plug, similar to that of the 2314, for each drive in the module. In addition, a CE service plug is supplied that is to be used when IBM-supplied diagnostic programs are executed.

Functionally, the 3330-series provides more capabilities than the 2314, particularly in the areas of performance and availability. The 3330-series supports all the standard 2314 commands (except the file scan commands) in addition to several new operations, including RPS and error recovery commands. (Table 20.10.3, at the end of this subsection, compares 3330-series and 2314 features.) The 3330-series also is an attractive growth device for the 2321 Data Cell Drive.

Table 20.10.1 compares the capacity and timing characteristics of the 3330-series with those of the 2314 and the 2321 Data Cell Drive. The increase in capacity achieved by replacing a 2314 or a 2321 with a 3330-series string depends upon the block size chosen for the data on the 3330. For example, if the 2314 full-track block size of 7294 bytes is maintained for a given data set on the 3330 to avoid programming changes, the 3330 yields a 91% increase in full pack capacity (almost twice the capacity). However, reblocking to a full track on the 3330, 13,030 bytes, yields a 242% full pack capacity increase. If not enough processor storage is available to allocate I/O areas of 13,030 bytes, lowering the 3330 block size to one-half of a 3330 track yields a 239% increase in full pack capacity.

If a 2321 is replaced by a 3330-series string, six full-track blocks of data from the 2321 (2000 bytes/2321 track) can be placed on each 3330 track, if full-track blocking is used, for a total of 92,112,000 bytes per 3336 pack (12,000 bytes times 7676 tracks per 3336 Model 1). Thus, slightly over four 3336 Model 1 packs provide the capacity equivalent of ten data cells, or a full 2321 drive, if full tracks are used. Ten full data cells, blocked full track, also can be contained in slightly more than four 3336 packs if half-track blocking is used on the 3336 Model 1.

Self-formatting records are written on 3336 packs the same as on 2316 packs. However, each physical area written (count, key, and data) has a field of error correction code appended to it for data validity checking by the 3830 or integrated storage control instead of the cyclic check area used on the 2314. The correction code used detects single burst errors of 22 bits or less and corrects single burst errors of 11 bits or less.

Table 20.10.1. Capacity and timing characteristics of 3330-series and 2314 disk storage and the 2321 Data Cell Drive

Characteristic	3330-series	2314	2321
Capacity in bytes truncated to the nearest thousand (full-track records)			
Pack or cell	100,018,000	29,176,000	39,200,000
String or Data Cell Drive			
2 drives/cells	200,036,000	58,352,000	78,400,000
4 drives/cells	400,073,000	116,704,000	156,800,000
6 drives/cells	600,109,000	175,056,000	235,200,000
8 drives/cells	800,146,000	233,408,000	313,600,000
10 cells	-	-	392,000,000
Access time (ms)			
Maximum	55	130	600 (for strip select and load)
Average	30	60	175 (minimum for strip select and load)
Average cylinder-to-cylinder	10	25	95 (on a strip)
Time channel busy searching when SET SECTOR is used (ms)			
Minimum	.120	-	-
Maximum	.380	-	-
Rotation time (ms)	16.7	25	50 (strip on drum)
Rotation speed (rpm)	3600	2400	1200
Data transfer rate (KB)	806	312	55

Table 20.10.2. 3336 and 2316 Disk Pack characteristics

Characteristic	3336	2316
Number of disks per pack	12	13
Number of recording disks	10	11
Number of recording surfaces (recorded tracks per pack)	19	20
Disk thickness in inches	.075	.050
Disk diameter in inches	14	14
Disk pack weight in pounds	20	15
Disk pack maximum capacity in millions of bytes	100	29.1
Full track capacity in bytes	13,030	7,294
Cylinders per pack	404 plus 7 alternates	200 plus 3 alternates
Tracks per cylinder	19	20
Tracks per pack	7,676	4,000

3830 STORAGE CONTROL

Model 1

The 3830 Storage Control Model 1 unit contains the power and the control functions required to operate one 3330-series string consisting of two, four, six, or eight drives. Only 3330 Disk Storage can be attached to 3830 Storage Control Model 1. When multiple requesting is used, the 3830 Model 1 can control concurrent operation of eight channel programs, one on each of drives. Only one of the eight drives can be transferring data at any given time. Figure 20.10.1 shows a 3330-series string attached to Model 1 of 3830 Storage Control.

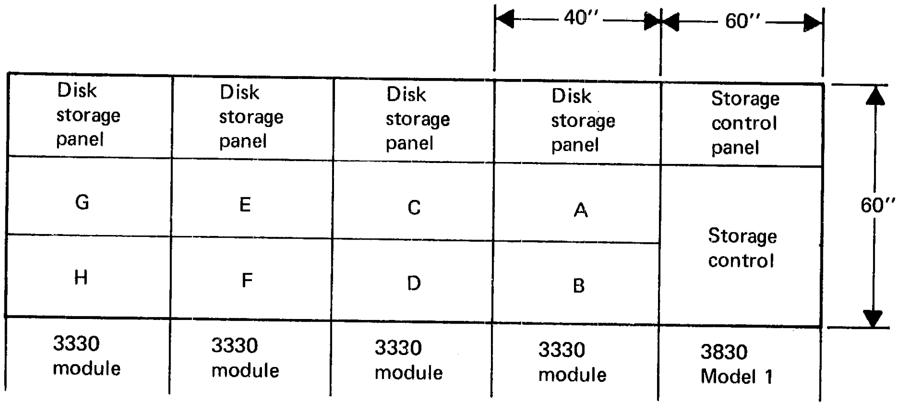


Figure 20.10.1. 3330-series disk storage attached to 3830 Storage Control Model 1.

The 3830 Model 1 control unit is microprogram controlled. Read/write monolithic storage contained in the control unit is used for microprogram residence. The control unit also contains a device that reads interchangeable disk cartridges (identical to the console file device). This device is used for microprogram backup storage and for storage of nonresident diagnostics for the 3330-series string. During a 3830 power-on sequence, the functional microprogram is loaded from the device into control storage within the control unit. Therefore, microcode engineering changes can be installed merely by replacing the current disk cartridge with another cartridge that contains the new microprogram.

The Two-Channel Switch feature, identical to the same feature for the 2314 facility, can be installed on a 3830 Model 1 control unit to allow the 3830 to be attached to two channels. The Two-Channel Switch, Additional feature can be added to this configuration to permit the 3830 to be attached to four channels. The four channels involved can be all on the same system, each on a different system, or some on the same system and others on different systems. The channels to which these two features are connected must have one control unit position and if block multiplexing is to be used, eight unshared subchannels available. The enable/disable switch can be set to dedicate the 3830 to any subset of the four channels.

The 3830 also incorporates new error detection, correction, and logging features, designed to improve its availability and serviceability. The following features implemented in the 3830 are not provided for previously announced 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

- Command retry initiated by the control unit to attempt hardware correction of certain errors without programming assistance

- Error logging by the control unit in its control storage of successful command retry operations

- Inline diagnostic tests contained on disk cartridges, which can be run on a single drive to diagnose hardware malfunctions while other drives in the string continue normal operations. (Inline diagnostics are provided only for 2314 facilities.)

Recovery of correctable data area errors. When the control unit detects a correctable data error during the reading of the data portion of a record, it generates the information necessary to correct the erroneous bytes. The sense bytes presented by the control unit 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.

Command retry. Error correction (without programming assistance) is performed by a channel/control unit command retry procedure without an intervening I/O interruption in the following situations:

1. When a correctable data error occurs during a search or read operation on home address, record count, or record key.

   During a search or read operation, the home address, count, or key read from the disk track is placed in a buffer in control storage within the control unit. When a correctable data error occurs, the control unit corrects the data in the buffer and reissues the command that caused the error. During reorientation to the record, the control unit disconnects and frees the block multiplexer channel. When the failing search or read command is reexecuted, the corrected data in the buffer is used instead of the data actually on the track.

2. When an uncorrectable data error is detected on any portion of the record during a read or a search operation.

The failing CCW is reissued twice by the control unit. If one of the two retries is successful, the channel program continues normally.

3. When a seek malfunction is detected.

   The control unit retries the command ten times in an attempt to position the access mechanism correctly.

4. When an alternate or defective track condition is recognized before data transfer begins.

   The control unit determines the location of the alternate or defective track (from R0 on the track), initiates a seek to this track, and orients to the index point. When this sequence completes, the original command is reissued by the control unit. This is a programmed procedure for previously announced System/360 direct access devices.

5. When a command overrun (or late command-chaining) condition occurs during execution of a channel program because of interference from another channel or the CPU.

   The control unit initiates a retry of the command that was late.

6. When a data overrun occurs, except for:

   a. A data overrun that occurs during a track overflow operation in the second or subsequent segments.

   b. A data overrun that occurs during a formatting write.

Error logging. Usage and error counters for each drive in the facility are maintained continuously in the control unit. The usage counters are used to accumulate the number of bytes read and seeks issued. The error counters are used to accumulate the number of seek, correctable data, and uncorrectable data errors that were retried successfully by a command retry procedure, as already described. Also accumulated is the total number of command and data overrun conditions that were retried by the storage control unit.

When a counter reaches its threshold or when a pack is removed from a drive, the control unit indicates the condition via a unit check when the next I/O operation is initiated to the drive. Counter data can be obtained and counters can be reset by issuing a SENSE or READ LOG command. These statistics can then be logged in the system error data set for later diagnosis.

Inline diagnostic tests. A 3830 Model 1 control unit can execute diagnostic tests on a malfunctioning drive while normal operations take place on the remaining drives in the string. After the service address plug is inserted in the malfunctioning drive, diagnostics can be executed on that drive by the customer engineer using the CE panel on the 3830 control unit. Operationally, the drive is offline to the control unit, and physically the drive is offline to the operating system.

Online testing of 3330-series drives can be performed under OLTEP control, as usual. Both OLT's and diagnostic programs contained in the OLT library can be executed on a malfunctioning drive via OLTEP. The diagnostic tests are loaded into control storage in the control unit from the OLT library. Operationally, the 3330-series drive is online to the control unit but is logically offline to the operating system.

Inline and online testing allows CE diagnosis and repair of most 3330-series drive failures without the necessity of taking the entire 3330-series string out of the system configuration.

Model 2

The 3830 Storage Control Model 2 unit is functionally equivalent to the 3830 Storage Control Model 1 unit, as the Model 1 is previously described, except for the following:

- Two strings of up to eight drives each can be attached to a channel via Model 2 of the 3830. When multiple requesting is used, the 3830 Model 2 can control up to 16 channel programs concurrently, one on each of its drives. Only one of the 2 to 16 drives attached to 3830 Model 2 can be transferring data at any given time.

- The 3830 Model 2 does not contain power for any of the 2 to 16 drives that can be attached to it.

- The 3830 Model 2 does not contain the device-oriented hardware that is present in Model 1 of the 3830.

- Inline diagnostics for a malfunctioning drive must be executed using the CE panel on the 3333 module.

Model 2 of the 3830 can have one or two strings of drives attached to it. Each string must include 3333 Disk Storage and Control as the first module attached to Model 2 of 3830 Storage Control. Up to three 3330 Disk Storage modules can then be added to each string for a total of eight drives per string. A 3333 module contains the power required for an eight-drive string and the device-oriented hardware that is not present in the 3830 Model 2. Only one 3333 module can be present in each string. A 3333 module is connected to a 3830 Model 2 via cables which can be a maximum of 150 feet in length. The 3830 Model 2 attaches to an integrated selector or block multiplexer channel in the Model 145 via a cable up to 150 feet in length. Model 1 of 3830 Storage Control can be field-converted to Model 2. Field conversion of Model 2 of 3830 Storage Control to Model 1 is not recommended. Figure 20.10.2 shows a Model 145 configuration with 3330-series disk storage attached via 3830 Storage Control Model 2.

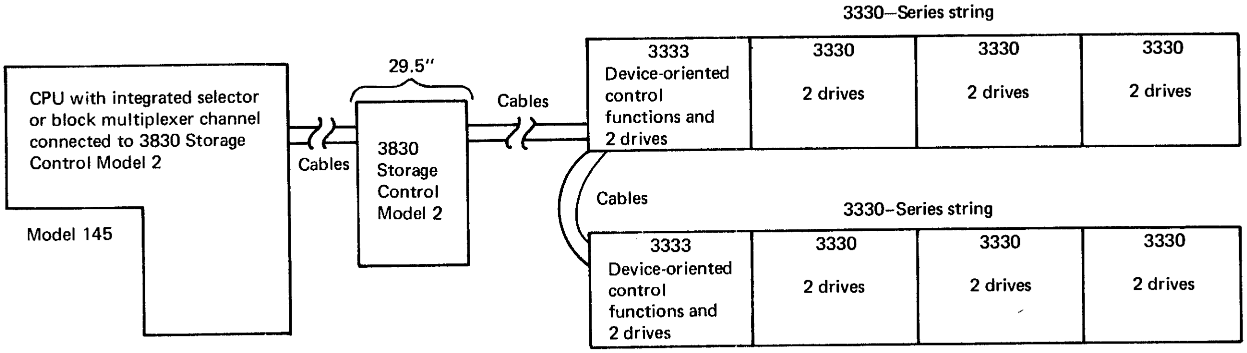

3330–Series string
CPU with integrated selector or block multiplexer channel connected to 3830 Storage Control Model 2
Model 145
Cables
29.5"
3830 Storage Control Model 2
Cables
3333 Device-oriented control functions and 2 drives
3330
2 drives
3330
2 drives
3330
2 drives
Cables
3330–Series string
3333 Device-oriented control functions and 2 drives
3330
2 drives
3330
2 drives
3330
2 drives

Figure 20.10.2. A Model 145 configuration with 3330-series disk storage attached via 3830 Storage Control Model 2

The Two-Channel Switch and the Two-Channel Switch, Additional features can be installed on a 3830 Model 2 unit to enable the strings it controls to be accessed by two or four channels, as discussed for the 3830 Model 1. The 3830 Model 2 provides lower cost attachment of two 3330-series strings to a channel than the 3830 Model 1, since only one 3830 Model 2 unit is required.

INTEGRATED STORAGE CONTROL

One or two strings of up to eight 3330-series drives each can be attached to a Model 145 selector or block multiplexer channel via integrated storage control (ISC) as well as via 3830 Storage Control. Each string consists of only one 3333 module and up to three 3330 modules. The integrated storage control unit supports the same functions and string configurations as 3830 Storage Control Model 2, except for the following:

- ISC is not a stand-alone unit and is contained in a 3345 Storage and Control Frame that is connected to a Model 145 CPU.
- The Two-Channel Switch, Additional feature (for four channel switching) cannot be attached to ISC.
- ISC is powered by the Model 145 CPU

A Model 145 configuration can include only one ISC unit, which must be assigned highest priority on the channel to which it is attached. Hence, a maximum of two strings of 3330-series drives can be attached to a Model 145 via ISC. The strings attached to a channel via ISC operate just as if they were attached to a channel via 3830 Storage Control Model 2. Figure 20.10.3 shows a Model 145 configuration with two 3330-series strings attached via ISC.

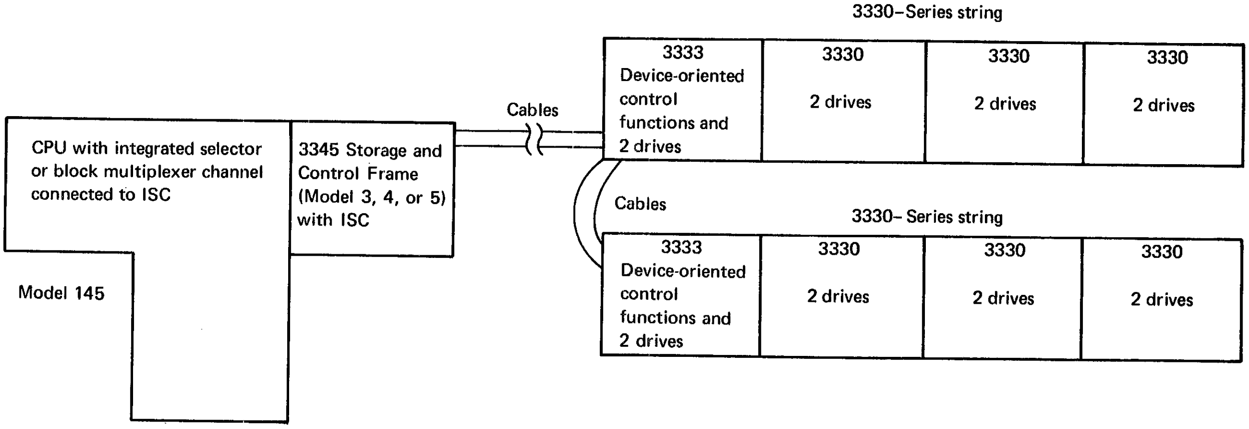

3330-Series string
3333 Device-oriented control functions and 2 drives
3330 2 drives
3330 2 drives
3330 2 drives
Cables
CPU with integrated selector or block multiplexer channel connected to ISC
3345 Storage and Control Frame (Model 3, 4, or 5) with ISC
Model 145
Cables
3330- Series string
3333 Device-oriented control functions and 2 drives
3330 2 drives
3330 2 drives
3330 2 drives

Figure 20.10.3. A Model 145 configuration with 3330-series disk storage attached via integrated storage control

The ISC unit is contained in a 3345 Storage and Control Frame. Listed below are the five models of the 3345 and their contents:

3345 Model	Contents
1	128K of processor storage only
2	256K of processor storage only
3	ISC only
4	ISC and 128K of processor storage
5	ISC and 256K of processor storage

Field conversion of 3345 models is provided as follows:

Model 1 to a Model 2, 4, or 5
Model 2 to a Model 5
Model 4 to a Model 5

Like 3830 Storage Control Model 2, ISC provides less expensive attachment of two 3330-series strings to a Model 145 channel than 3830 Storage Control Model 1. For Model 145 configurations with more than 256K of processor storage, less floor space is required when a 3330-series string is attached via ISC instead of via 3830 Storage Control Model 2, since the ISC unit is contained in the 3345 Storage and Control Frame.

Table 20.10.3. Hardware features of 3330-series and 2314 disk storage. Features are available for 3330-series strings attached via 3830 Storage Control Models 1 and 2 and via Integrated Storage Control unless otherwise indicated.

Feature	3330-series	2314
Number of drives per string or facility	2,4,6, or 8	1,2,3,4,5,6,7, or 8 (A ninth can be included as a spare only.)
Removable interchangeable disk packs	Yes	Yes
Removable address plugs	Yes	Yes
Record Overflow feature	Standard	Standard
File Scan feature	Not available	Standard
Multiple track operations	Standard	Standard
Two-Channel Switch	Optional	Optional
Attachment of 3830 to four channels	Yes, using the Two-Channel Switch and Two-Channel Switch, Additional features	Yes, using Two-Channel Switch and 2844 Auxiliary Storage Control
Second control unit (to permit two concurrent data transfer operations on a string or facility)	Not available	Optional (2844 Auxiliary Storage Control)
Rotational position sensing	Standard (128 sectors/track)	Not available
Multiple requesting	Standard	Not available
Command retry by control unit and channel	Standard	Not implemented
Error correction data presented by control unit	Yes	No
Writable storage in control unit loaded from a disk cartridge	Yes	No
Inline diagnostic tests initiated via the CE panel in the string or facility	Standard	Standard
Inline diagnostic tests initiated via the system console	Standard	Not implemented

SUMMARY

The 3330-series offers more than large capacity, faster access, and attractive price performance. A 3330-series string and its control unit are actually a subsystem. A 3830 control unit or integrated storage control can control the concurrent execution of one RPS channel program on each of its drives and can handle certain error correction and logging functions, which normally must be programmed, thereby relieving the control program of these activities. In addition, the availability and serviceability of the 3330-series are improved by the implementation of error correction features in hardware, by use of inline diagnostics, and by the speed and ease of engineering change installation. These factors add to the improvement of total system availability.

20:15 THE 2305 FIXED HEAD STORAGE MODULE AND 2835 STORAGE CONTROL MODEL 2

One or two 2305 Fixed Head Storage Modules can be attached to 2835 Storage Control. Each module contains six nonremovable rotating disks on which data is recorded. Read/write heads, called recording elements, are fixed in position to access each track on the twelve recording surfaces so that no arm motion is required. (See Tables 20.15.1 through 20.15.3 at the end of this subsection for a comparison of 2305 Model 2 and 2303 Drum Storage characteristics and capacities.)

Spare, or alternate, tracks are provided in 2305 modules and must be wired in by a customer engineer to replace defective recording tracks. However, one spare track is available for assignment by the alternate track assignment utility program when a permanent track error occurs on a recording track during processing. Once a spare has been assigned as an alternate track, the hardware automatically accesses the alternate track when the defective recording track is addressed. This is called alternate track sparing. Switching to an assigned alternate track during processing is a programmed action for currently announced direct access devices.

The 2835 control unit provides new error correction facilities similar to those of the 3830 control unit. Recorded data areas within self-formatting records have ten correction code bytes appended to them instead of a two-byte cyclic check code. When certain types of data errors occur during the reading of the data portion of a record, the control unit can determine the bits in error and generate correction data. This recovery information is presented to the error routine via the sense bytes and can be used to correct the invalid record in processor storage (as described for 3330-series disk storage).

A command retry feature is implemented in the 2835. This feature permits certain types of failing commands to be reissued automatically by the channel, when requested by the control unit, without an intervening I/O interruption. For example, when a count or key area is read erroneously, the control unit retries the command once. If the error is not corrected by the retry, the control unit corrects the data in its own buffer, reexecutes the failing read, and presents the corrected data from the buffer instead of reading it from the track.

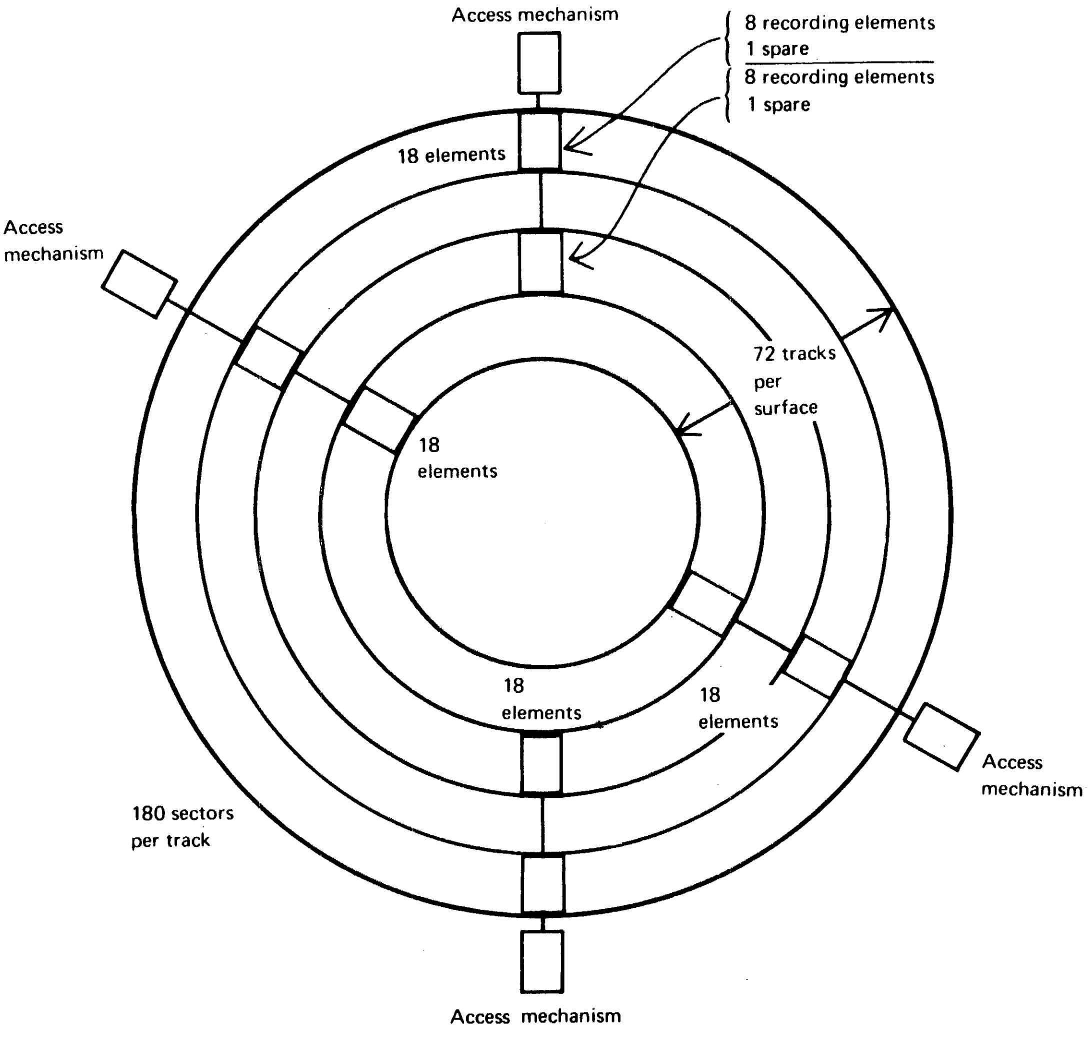
Like the 3830 control unit, the 2835 contains a device that reads disk cartridges containing the control unit microprogram and diagnostic routines.

DATA RECORDING

Data tracks on the 2305 Model 2 facility are formatted in the same manner as on System/360 direct access devices, except for the absence of a home address on each track. There are 768 recording tracks and 96 spare tracks in one module. One recording element is positioned over each track. Each of the twelve surfaces contains 72 tracks, 64 recording and 8 spare. The spare tracks are interspersed among the 72 tracks so that every ninth track is a spare. Data is recorded serially by bit on each track.

Four nonmovable access mechanisms are positioned around the rotating disks as shown in Figure 20.15.1. Each access mechanism contains two groups of 9 recording elements per surface (for a total of 16 recording and 2 spare elements) and accesses one-quarter of the tracks on each surface. A group of 8 recording elements accesses every other track. The outermost element group of the access mechanism at the top of Figure 20.15.1 accesses data tracks 1, 3, 5, ..., 15, while data tracks 2, 4, 6, ..., 16 are accessed by the outermost element group of the access mechanism at the bottom of the diagram.

There are 180 sectors per track on the 2305 Model 2. When RPS is used, the search time, from sector found to beginning of desired record, ranges from a minimum of 112 microseconds to a maximum of 167 microseconds.



64 recording tracks/surface x 12 = 768 recording tracks/module
8 spare tracks/surface x 12 = 96 spare tracks/module
216 recording elements (read/write heads)/access mechanism x 4 = 864 recording elements

Figure 20.15.1. Top view of a 2305 Model 2 disk surface

ROTATIONAL POSITION SENSING AND MULTIPLE REQUESTING

RPS is a standard feature of the 2835 control unit as is the other new capability called multiple requesting, which allows up to eight channel programs to be active concurrently on each of the two 2305 modules that can be attached to the control unit. In other words, a 2305 module can be viewed as eight logical devices, although physically it is only one device.

As described previously, rotational position sensing and block multiplexing permit a direct access device to disconnect during set sector operations. These facilities, used in conjunction with the multiple requesting feature, permit concurrent operations to take place on each 2305 module in a facility. Thus, the effective data rate of a module can be increased substantially.

The multiple requesting capability is implemented by associating eight logical device addresses (0-7) with a 2305 module. Each logical device address is also assigned to a specific subchannel of a block multiplexer channel and a specific register (0-7) in the 2835 control unit. When a channel program is initiated, it is associated with an available logical device address by data management (I/O supervisor). (Logical device addresses are not permanently assigned to specific tracks or data sets in a module.) When the SET SECTOR command is issued, its specified sector number is stored in the register in the control unit that is assigned to the logical device address being used for the channel program. Then the control unit disconnects from the channel.

At this point, another channel program with a SET SECTOR command can be accepted by the channel and control unit (assuming neither is busy). This channel program will be initiated using another available logical device address and its assigned control unit register. This process can be repeated for up to eight SET SECTOR commands, so that eight channel programs can be executing concurrently per 2305 module.

Whenever the control unit is not executing a command or is not otherwise busy, it monitors the rotational position counter in the 2305 module that is being incremented each sector time period. When the sector number in the counter of a module compares equal with the sector number stored in one of the registers in the control unit and the channel is free, the control unit reconnects and resumes execution of the suspended channel program associated with the logical device address assigned to the control unit register that compared equal (see Figure 20.15.2).

It should be noted that one 2305 module requires eight logical device addresses, each of which requires one subchannel on a block multiplexer channel. Since a 2835 control unit can have two modules, one 2305 facility can use 16 device addresses and 16 nonshared block multiplexer subchannels.

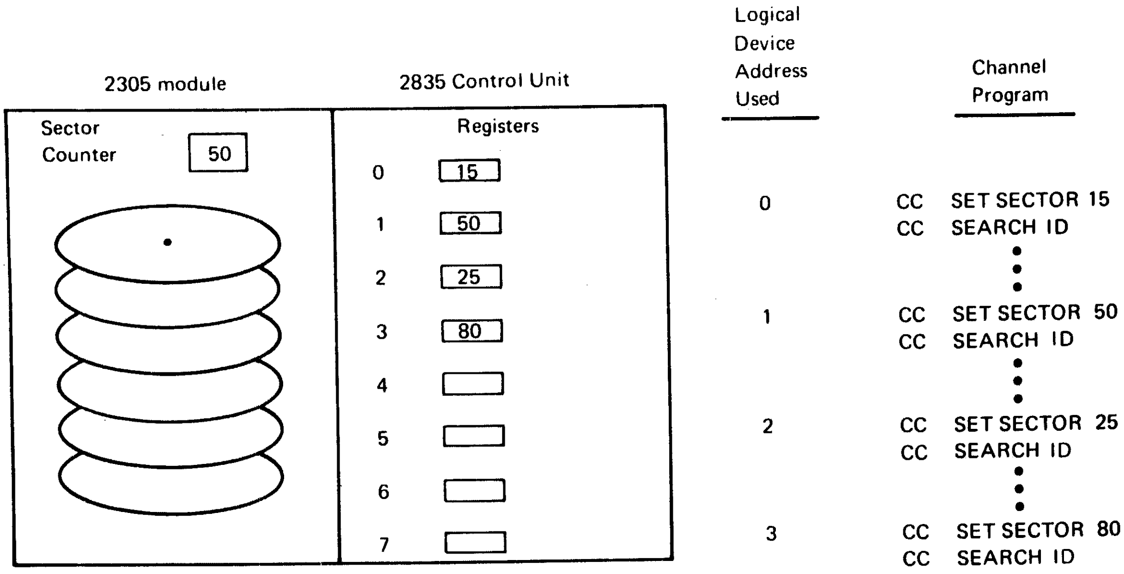


Figure 20.15.2. Multiple requesting on the 2305 facility

Table 20.15.1. 2305 Model 2 facility and 2303 Drum Storage characteristics

Characteristic	2305 Model 2 Module	2303 Drum Storage
Device type	Six rotating disks with twelve recording surfaces	Rotating drum
Module capacity in bytes (full-track records, no key)	11,258,880	3,913,000
Number of recording tracks	768	800
Number of bytes per track (R0, R1 written without key)	14,660 Home address is never written on a track	4,892 Home address is always written on each track
Number of read/write heads (recording elements) per module	864 One positioned to access each of 768 recording and 96 spare tracks	880 One positioned to access each of 800 recording and 80 alternate tracks
Rotation time (ms)	10	17.5
Access time (ms)		
Maximum	10.25	17.5
Average	5.0	8.6

Table 20.15.1. (continued)

Characteristic	2305 Model 2 Module	2303 Drum Storage
Time channel busy searching when SET SECTOR is used (ms)		
Minimum	.112	---
Maximum	.167	---
Rotation speed (rpm)	6000	3400
Data transfer rate (MB)	1.5	.303
Data validity checking	10 correction code bytes (CCB) appended to each area written	Two-byte cyclic check (CC) code appended to each area written
Error recovery performed by the control unit	1. Command retry to retry a failing command without an I/O interruption 2. Correction of data errors that occur in the data area of a record is possible by programming using information in sense bytes	Not provided
Rotational position sensing	Standard feature (180 sectors per track)	Not available
Multiple requesting (allows concurrent I/O operations on one module)	Up to 8 concurrent operations per 2305 module (up to 7 per module supported by data management)	One I/O operation at a time is supported.
Record overflow	Standard feature	Optional feature
Two-Channel Switch	Optional feature	Optional feature
Device contained in the control unit	Yes, to read the control unit microprogram and diagnostic programs	No

Table 20.15.2. Effective capacity of the 2305 Model 2 and the 2303 for various block sizes (DL column) with a 25-byte key

KL = 25	Effective Capacity in Bytes	
DL in Bytes	2305 Model 2	2303
100	2,688,000	1,200,000
200	4,300,800	1,760,000
300	5,529,600	1,920,000
400	6,144,000	2,240,000
500	6,912,000	2,400,000
600	7,372,800	2,400,000
700	7,526,400	2,240,000
800	7,987,200	2,560,000
900	8,294,400	2,160,000
1000	8,448,000	2,400,000
2000	9,216,000	1,600,000
3000	9,216,000	2,400,000
4000	9,216,000	3,200,000

Table 20.15.3. Effective capacity of the 2305 Model 2 and the 2303 for various block sizes (DL column) when records are written without key

KL = 0	Effective Capacity in Bytes	
DL in Bytes	2305 Model 2	2303
100	3,763,200	1,600,000
200	5,683,200	2,080,000
300	6,681,600	2,400,000
400	7,372,800	2,560,000
500	8,064,000	2,400,000
600	8,294,400	2,400,000
700	8,601,600	2,800,000
800	8,601,600	2,560,000
900	8,895,600	2,880,000
1000	9,216,000	2,400,000
2000	9,216,000	1,600,000
3000	9,216,000	2,400,000
4000	9,216,000	3,200,000

20:20 THE 3211 PRINTER

The 3211 is a high-speed line printer with front printing and new features designed to reduce operator intervention. The 3211 can print 2000 alphameric lines per minute (with a 48-character set) and is designed to be used in any installation that has a high volume of print activity. The 3211 attaches to all System/370 models and System/360 Models 30 and up.

The 3211 has a standard 132-print-position line, which can be expanded to 150 positions as an option. The number of print positions does not affect printing speed. The Universal Character Set (UCS) feature is standard, and the 3216 Interchangeable Train Cartridge contains 432 graphics. The cartridge character arrangement is unrestricted and can be alphabetic, numeric, or special characters in any combination. When the character arrangement is optimized for specific printing loads, speeds of up to 2500 lines per minute can be attained.

The 3211 can also be used to print documents that can be read by the 1287 Optical Reader and the 1288 Optical Page Reader. When the optional OCR Print Package is installed on the 3211 Printer, one of two available 3216 OCR cartridges, each with 48 graphics, can be used.

The 3211 attaches to a 3811 control unit. Unlike some models of the 2821 control unit, which can handle multiple devices, a 3811 controls only one 3211 Printer. The commands used for a 1403 printer are a compatible subset of those provided for the 3211 printer. Print, skip, and space commands for the two printers are identical.

New features of the 3211 include a powered forms stacker, an automatic platen, and a tapeless carriage. The powered stacker mechanism is self-adjusting and automatically rises in increments as the stack of paper mounts. This ensures that the stacker mechanism is always the same distance above the top of the stack of forms. The rate of rise during each increment is determined by the setting of the stacker rate knob, which can be adjusted by the operator to produce the best condition for the thickness of the forms being stacked. The stacker also can be raised or lowered manually.

When forms are inserted, the printer platen automatically positions itself close to the train cartridge in accordance with the thickness of the forms. Thus, correct clearance between the platen and the cartridge is achieved without operator intervention. Because of its automatic forms thickness sensing, the 3211 is sensitive to forms with a different degree of thickness at each edge. (For forms limitations, see IBM 3211 Printer, 3216 Interchangeable Train Cartridge, and 3811 Printer Control Unit Component Description and Operator's Guide, GA24-3543.)

Forms control paper carriage tape loading and unloading by the operator is eliminated by implementation of a tapeless carriage feature for the 3211. Forms spacing and skipping are controlled by a program-loaded forms control buffer (FCB) contained in the 3811 control unit.

The FCB contains 180 storage positions, each of which corresponds to a print line, that is, a single space of the carriage. Thus, forms up to 22.5 inches in length can be accommodated at eight lines per inch spacing (or 24 inches at six lines per inch). Up to twelve channel codes (1-12), corresponding to the twelve channel positions of the paper carriage tape used on a 1403 Printer, can be stored in the appropriate buffer line positions to control carriage skipping. The FCB can be considered to contain a storage image of a carriage control tape.

A carriage control address register is used to address the FCB and maintain correct line position with respect to the form. This register is incremented as space and skip commands, which cause the form to advance, are issued. When a SKIP TO CHANNEL command is executed, the carriage control address register is incremented and the form moves until the channel specified is sensed in a line position in buffer storage. If the requested channel number is not found in the buffer, forms movement stops after address position 1 (line 1) has been sensed twice. This prevents runaway forms skipping.

A flag in a buffer storage line position is used to indicate the last line of the form for forms shorter than 180 lines. A flag bit is also used in the first buffer storage position to indicate six or eight lines per inch spacing. The FCB is loaded with the desired forms spacing characters via a LOAD FCB command issued by a program. An error indication is given if an end-of-page flag is not present or if an invalid carriage code is loaded.

The 3811 includes a new indexing capability as a standard feature. With this facility, a print line can be offset to the right by up to 30 print positions (3 inches) without operator intervention or program alteration. Hence, a program written to print in positions 1 to 80, for example, could print 80 characters beginning with any position from 1 to 31 without adjusting the printer or changing the program. The offset, 0 to 30 positions, is indicated in the print position indexing byte in the forms control buffer. This byte is loaded using the LOAD FCB command that also loads forms control characters.

Serviceability features, in addition to those provided for the 1403 Printer, are incorporated into the design of the 3211. The fact that a 3811 control unit controls only one 3211 Printer, instead of multiple devices, permits offline repair of the malfunctioning printer or control unit only, without the necessity of removing other operational units from the system.

The 3811 control unit presents six bytes of sense information to identify printer and control unit malfunctions instead of only one byte, as is provided for the 1403. Certain errors (such as a parity check in the print line buffer) that might be corrected by programmed retry of the print operation are identified in the sense bytes, and carriage motion is suppressed. This permits error recovery without operator intervention if the retry is successful. The additional status data presented can be stored for later analysis and should speed the diagnosis of hardware malfunctions.

20:25 THE 3803/3420 MAGNETIC TAPE SUBSYSTEM

The new 3803/3420 Magnetic Tape Subsystem consists of 3803 Tape Control and a family of three 3420 Magnetic Tape Units, which read and write nine-track, 1600-BPI, phase-encoded, half-inch magnetic tape. The three tape units, Models 3, 5, and 7, have a data rate of 120 KB, 200 KB, and 320 KB, respectively, and up to eight tape units, in any mixture of models, can be attached to a 3803 control unit. This tape subsystem, which embodies a completely new control unit technology, offers price performance improvements, compatibility with existing seven- and nine-track tape volumes and programs, enhanced reliability, availability, and serviceability features, lower cost tape-switching capabilities, and standard automated tape-handling features presently available only on 2420 Magnetic Tape Units. (Table 20.25.2 at the end of this subsection compares 3420 and 2401 tape unit characteristics.)

The 3803/3420 subsystem can be attached to all System/370 models and to System/360 Models 30 to 195, except Model 44 for which there is no program support. The tape commands, status responses, and basic sense data of this tape subsystem are compatible with those of 2400-series tape units. Thus, any correctly written, non-time-dependent System/360 program for 2400-series tape units will operate without change on the Model 145 (subject to restrictions stated in Section 10:05) to handle operations on 3803/3420 subsystems with equivalent features installed. That is, existing nine-track 1600-BPI phase-encoded (PE), nine-track 800-BPI non-return-to-zero (NRZI), and seven-track 556/800-BPI NRZI-encoded tapes can be processed on 3420 tape units using existing programs without change to the tape volumes or programs.

The 3803/3420 tape subsystem offers users with intermediate systems the advantages of the latest significant advances in tape speed and design while maintaining media compatibility with existing tape volumes and providing enhanced RAS features. Specifically, the following are provided:

- Data rates of 120 KB, 200 KB, and 320 KB at 1600-BPI density
- Phase-encoded data recording that automatically detects and corrects single-bit read errors in-flight

- A tape transport design that minimizes tape wear and increases reliability, a single-capstan drive to control tape movement that provides faster data access times and rewinds, and more precise control of motor speed to help minimize damage to tape media

- Cartridge loading of tape, automatic tape threading, and a new automatic tape reel hub latch, all to reduce tape setup time

- Dual Density and Seven Track (mutually exclusive) features to enable a 3420 tape unit to handle either nine-track 800-BPI NRZI and 1600-BPI PE tape or seven-track 556/800-BPI NRZI (BCD or binary) tape

- Flexible, lower cost tape switching implemented in a new compact physical design. A two-channel switch is available also.

- Features such as new technology to improve subsystem reliability and new diagnostic facilities to aid serviceability and thereby increase subsystem availability

Phase-encoded recording. The phase-encoded (PE) recording technique offers superior error detection and reliability as compared with the non-return-to-zero (NRZI) technique. In both cases, magnetic recording of one and zero bits is accomplished by means of flux reversals or changes in polarity. In NRZI recording, only one bits are recorded as magnetized spots, and a flux reversal occurs only for one bits. In PE recording both zero and one bits are recorded (the zero bit and one bit being opposite in polarity), and a flux reversal is required in every bit position. Thus, the PE dual flux recording technique differentiates between no recording and the presence of a zero bit, and the absence of any signal is detected as an error.

The positive recording of all zero and one bits in PE eliminates the need for horizontal parity bits (longitudinal redundancy check used in NRZI recording), and vertical parity bits are used to correct single-bit read errors in flight. During reading, if a single track fails to respond with a suitable pulse in any bit position, reading of the rest of that track is immediately disabled for the remainder of the data block, and the remaining bits for that track are automatically generated by use of the vertical parity bits. In-flight single-track error correction eliminates the time normally lost in backspacing and rereading NRZI tape for correction of single-track dropouts or defects.

Phase encoding offers other advantages. If a string of zeros is recorded on tape, successful reading in NRZI requires close synchronization to "count" the correct number of zeros. With PE, this synchronization is provided by the flux reversal in every bit position; hence, PE recording (and reading) is self-clocking. In addition, each block written on a PE tape is preceded and followed by a coded burst of bits in all tracks to set up the individual track-clocking rates. The read circuitry is designed to recognize these bursts and thereby minimize the effect of noise in the gap.

The critical nature of vertical skew (alignment of bits within a byte) that is imposed by NRZI recording is minimized by this individual track-clocking scheme (one clock per track versus one clock for the entire tape subsystem), and by the use of one-byte (nine-bit) capacity skew buffers that can be in the process of collecting up to four data bytes at the same time, as the tape passes the read head. Because of the positive recording of all bits, once a skew buffer contains nine bits, one from each horizontal data track, it is an indication that a byte has been read. Thus, the 3420 can handle the situation in which the tape is not exactly aligned, and bits from up to four adjacent bytes can be read concurrently.

Like 2400-series tape units, the 3420 utilizes a two-gap read/write head that performs readback checking during write operations. The 3420 also has a separate erase head that erases the entire width of the tape during any write operation before writing occurs. Full-width erasure reduces the likelihood of leaving extraneous bits in interblock gaps or skip areas and minimizes the interchangeability problems that can occur when tape is written on one tape unit and read on another.

Advanced engineering design. The tape path in the 3420 tape unit is designed for "soft handling" of tape volumes to minimize tape wear and thus improve tape reliability. Other features, such as the single-capstan drive and optical tachometers, result in faster data access and rewind times than those of the 2401.

On a 3420 tape unit, the tape reel is mounted on the right side of the tape transport, instead of on the left as on a 2401 tape unit, so that an inverted tape path exists. As a result, when the tape is loaded in the columns, the recording side touches only the tape cleaner and read/write head. Friction and tape wear are also reduced by the presence of air bearings in the tape transport that provide a thin film of air between the nonrecording surface and each metal bearing.

Use of a single-capstan drive transport for tape movement and optical tachometers for control of motor speed result in several advantages. First, faster access times than those of 2401 tape unit models are achieved. Access time is defined as the time interval from initiation of a write or forward read command (given when the tape is not at load point) until the first data byte is read or written, assuming the tape is brought up to speed from stopped status. Nominal access times for 3420 Models 3, 5, and 7 are 4.0 ms, 2.9 ms, and 2.0 ms, respectively.

Second, the single-capstan drive can be made to operate faster than normal read/write speed, and in-column rewind is thus implemented. Full reel rewind speeds average 410, 480, and 640 inches per second for Models 3, 5, and 7, respectively. In addition, less time is required to rewind less than a full reel on a 3420 as compared to a 2401 because of faster rewind times achieved by in-column rewinding.

Last, three optical tachometers that monitor motor speed are used to achieve precise control of the speed of both the capstan motor and the tape reel motors. The capstan tachometer measures the size of the interblock gaps (IBG's) created during tape writing. The result is a more consistent IBG size (.6 inches) than is created by 2400-series tape units, which enables more accurate calculation of tape passing time. IBG passing times are 8.0 ms, 4.8 ms, and 3.0 ms for 3420 Models 3, 5, and 7, respectively. These times would be used in calculations for command-chained tape operations (reading or writing more than one tape block with a single START I/O instruction). More precise capstan motor speed also results in smoother starts and stops, thereby minimizing tape stretching and breaking.

The two tape reel tachometers measure tape speed as the tape enters and leaves the vacuum columns, and tape speed is adjusted when necessary. The 3420 tape unit is, therefore, less sensitive to voltage changes. More precise control of tape reel motor speed improves rewind speed and minimizes erratic tape stacking during rewinds so that there is less chance of damaging tape edges.

Automatic threading and cartridge loading. These advanced features are standard on all 3420 models and significantly reduce tape mounting and demounting time. Tape threading is automatic for tape reels not enclosed in a wraparound cartridge once the reel (10.5-inch, 8.5-inch, or minireel) is mounted on the tape unit with the tape end placed in the threading chute and the load-rewind button is depressed. The power window is closed, the tape is threaded on the takeup reel, and the tape is loaded in the columns and positioned at load point within ten seconds after the button is depressed for Models 3 and 5. On the Model 7, only seven seconds are required. In addition, unload and rewind/unload operations cause the tape to be completely rewound on the tape reel and the power window to be lowered so that the reel is ready for immediate demounting.

If the tape is enclosed in a wraparound cartridge (10.5-inch reels only), an operator need only mount the cartridge and does not have to place the tape end on the threader chute. Once the load-rewind button is depressed, ten seconds are required to open the cartridge and perform automatic threading. If automatic threading fails on the first try, the 3420 unit automatically rewinds the tape and retries threading. Unload operations rewind and close the cartridge automatically. In addition to fast tape reel mounting, the use of a wraparound cartridge offers other advantages. Handling of the tape reel itself is not required when the tape is used, because the wraparound cartridge is also the shelf storage container. The only time the cartridge need be opened is when it is opened by the 3420 during use. This enhances the reliability of the tape media.

The 3420 tape unit also has a new automatic reel latch instead of the snap-type hub latch implemented on newer 2400-series tape units. The operator places the tape reel on the hub and the automatic latch mechanically aligns the reel and then pneumatically locks it in position.

The advantage of these features can be shown by comparing setup times for tape units with and without the autothread feature. A tape study using experienced operators indicated the total time required to remove a tape reel, mount a new reel, thread the tape, and come to ready status was the following:

2401 tape unit - 40 seconds
Autothread tape unit without cartridge - 29 seconds
Autothread tape unit with cartridge - 13 seconds

Single Density, Dual Density, and Seven Track features. These three features are provided for both 3803 control units and 3420 tape units. They are mutally exclusive features. The Dual Density or the Seven Track NRZI feature can be field installed on a 3420 tape unit only if it is replacing another NRZI feature. (For example, Dual Density can be field installed to replace the Seven Track but not the Single Density feature.) The Dual Density and Seven Track features facilitate efficient conversion of existing NRZI-recorded tapes to 1600-BPI phase-encoded format and permit tape volume interchange with other systems that use seven-track 556/800-BPI or nine-track 800-BPI tape. (See Section 60 for a discussion of conversion to 3420 tape units.)

A 3803 control unit with the Single Density feature (without a switching feature) can handle up to eight 3420 tape units (Models 3, 5, and 7) with the companion Single Density feature installed. Only 1600-BPI PE tape can be read and written. When the Dual Density feature is present on the 3803 control unit, both nine-track 1600-BPI PE and nine-track 800-BPI NRZI tape operations can be performed on 3420 units (Models 3, 5, and 7) with the companion Dual Density feature installed. (Tape units with the Single Density feature still handle only nine-track 1600-BPI PE tape.) When the Seven Track feature is present on the 3803 control unit, seven-track 556/800-BPI NRZI operations (both BCD and binary format) can be performed on 3420 tape units (Models 3, 5, and 7) with the companion Seven Track feature installed. The data convert and translate facilities are a standard part of the Seven Track feature. Table 20.25.1 summarizes 3803 control unit capabilities without and with these features.

Tape mode setting is handled as follows. For write operations on nine-track tape units with the Dual Density feature, a mode set command must be issued to establish 1600-BPI PE or 800-BPI NRZI recording mode prior to the first write. Tapes written in PE mode have a format identification burst recorded at load point that differentiates them from NRZI mode tapes. During reading, sensing of this burst automatically puts the tape unit in PE mode. Failure to sense the burst establishes NRZI mode if both the tape unit and control unit have the Dual Density feature. If an attempt is made to read NRZI-mode tape on a unit without the Dual Density feature, an error indication results. Once PE or NRZI mode is established for read operations, it is retained until the tape returns to load point.

For seven-track read and write operations, NRZI mode, density, parity, and use of the data converter or translator are established by issuing a single MODE SET command.

Table 20.25.1. 3803 control unit configurations and capabilities with Single Density, Dual Density and Seven Track features

3803 with Single Density Feature	3803 with Dual Density Feature	3803 with Seven Track Feature (includes data convert and translate)
1. Nine-track, 1600-BPI PE tape on 3420 Models 3, 5, and 7 with Single Density feature	1. Nine-Track, 1600-BPI PE tape on 3420 Models 3, 5, and 7 with Single Density feature 2. Nine-track, 800-BPI NRZI tapes and nine-track, 1600-BPI PE tapes on 3420 Models 3, 5, and 7 with the Dual Density feature.	1. Nine-Track, 1600-BPI PE tape on 3420 Models 3, 5, and 7 with Single Density feature 2. Seven-track, 556/800-BPI, NRZI BCD and binary tapes on 3420 Models 3, 5, and 7 with the Seven Track feature.
Note: The Single Density, Dual Density, and Seven Track features are mutually exclusive on the same control unit or the same tape unit.		

Tape-switching features. Tape subsystem configuration flexibility is provided by field installable tape-switching options that permit up to four control units to be switched among up to 16 tape units. While this capability is provided for 2400-series tape units via the 2816 Switching Unit, tape switching for the 3803/3420 subsystem offers the advantages of compact design, reduced cost, and enhanced subsystem availability.

The switching features are built into the 3803 control unit itself so that space for stand-alone switching units is not required. The fact that tape-switching features are contained in the 3803 control units being switched (rather than in one unit) also enhances tape subsystem availability. When a switch failure occurs in one control unit, that unit can be switched offline, eliminating the necessity of removing the entire tape-switching subsystem from the operative system configuration.

Using combinations of the Communicator and the Two-Control Switch, the Three-Control Switch, or the Four-Control Switch optional features, two, three, or four control units can be configured to be switched among up to 8 or up to 16 tape units. The Communicator must be present in all control units that are to be switched. It allows the control unit in which it is installed to address tape units that are attached to an interconnected control unit. Figure 20.25.1 shows the switching feature requirements for permissible switching combinations. The switch combinations shown for switching control units among up to 16 tape units are the same that are required for switching control units among up to eight tape units.

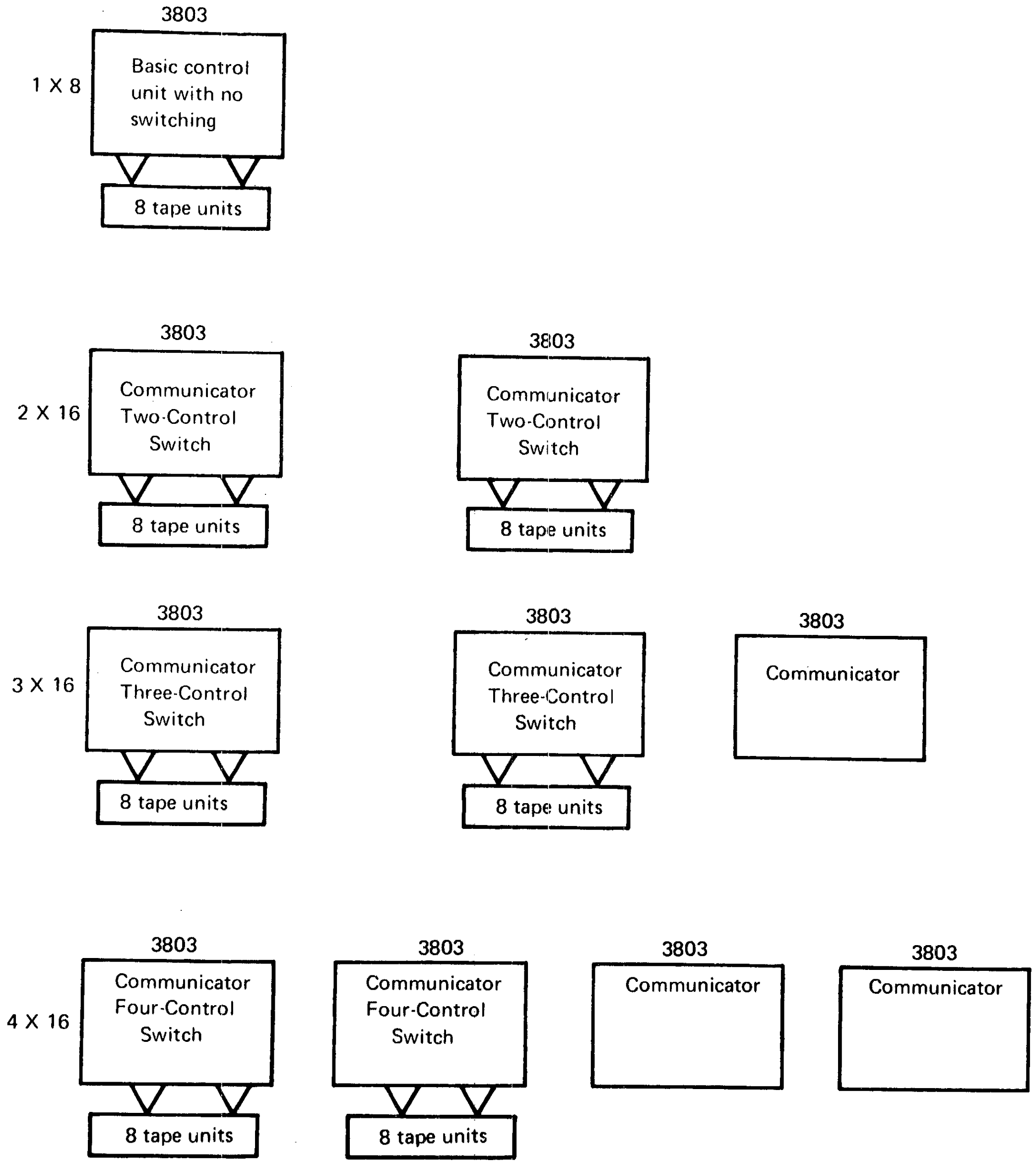

3803
1 X 8
Basic control unit with no switching
8 tape units
3803
2 X 16
Communicator Two-Control Switch
8 tape units
3803
Communicator Two-Control Switch
8 tape units
3803
3 X 16
Communicator Three-Control Switch
8 tape units
3803
Communicator Three-Control Switch
8 tape units
3803
Communicator
3803
4 X 16
Communicator Four-Control Switch
8 tape units
3803
Communicator Four-Control Switch
8 tape units
3803
Communicator
3803
Communicator

Figure 20.25.1. Tape-switching configurations for the 3803/3420 Magnetic Tape Subsystem

A two-control-unit switching configuration is required to replace the 2804 and 2404 read-while-write control units. The advantage of the tape-switching approach is that for a small price increment better performance is possible. This is true because any two tape operations can be active concurrently in a switched configuration (including two reads or two writes), while the degree of simultaneity achieved using a read-while-write control unit is application dependent. That is, the application must lend itself to reading, then writing (or vice versa).

Two-Channel Switch. A 3803 control unit with the two-channel switch installed can be attached to two channels in the same system or in two different systems. This feature can be present on a 3803 that also has tape-switching features installed.

When the two-channel switch is installed on a 3803, the switch can be set to permit only one channel or the other to access the control unit and its tapes, or the switch can be set to allow access to the control unit and its tapes by both channels, one channel at a time. In the latter case, if channel A requests an operation when the control unit is busy performing an operation on channel B, channel A must wait until the control unit becomes available again.

The two-channel switch can be used to connect a 3803 to two channels in the same system to provide the capability of switching the tapes from one channel to another or to provide two channel paths to the tapes connected to the 3803.

Similarly, the two-channel switch can be used to connect a 3803 to two channels in two different systems to provide the capability of switching all the tapes from one system to the other, for backup purposes. Alternatively, the 3803 can be set to allow access to it by both systems. In this case, each tape connected to the 3803 can be accessed by only one CPU. Those that are to be accessed by CPU A must be varied offline to CPU B, and vice versa, by the operator. Partitioning of the tapes in a two-system environment is strictly the responsibility of the operator. There are no physical controls on a 3803 with a two-channel switch that provide for partitioning the tapes between the two CPU's, nor is there any programming systems support that checks whether or not a tape is enabled to both systems.

The two-channel switch for the 3803/3420 subsystem offers configuration flexibility not generally available to 2400-series tape unit users. A two-channel switch currently is provided only for a 2803 Model 1 control unit and can be used only in Model 67 and in Model 65 multiprocessing configurations.

Data security erase command. When a data security erase command is issued, the tape unit selected erases tape from the point at which the operation is initiated until the tape indicate (end-of-file) marker is sensed. This command is not provided for 2400-series tape drives.

Reliability, availability, and serviceability features. The 3803/3420 hardware subsystem has several RAS features, in addition to the reliability and availability features already discussed for the tape media itself.

The 3803 control unit embodies a totally new design. The newest monolithic logic technology is used in the 3803 control unit, and it therefore offers greater reliability and more compact physical design in comparison with the 2803 control unit. (The 3803 is approximately half the size of a 2803 control unit.) In addition, both logic circuitry and mechanical components in the control and tape units are functionally packaged to enable more rapid fault location and faster replacement.

As a diagnostic aid, additional sense bytes are generated by the microprogram-controlled 3803 control unit. The 3803 uses ROS for microprogram residence. Twenty-four sense bytes are provided, instead of the six generated by the 2803, certain of which can be used in tracing control unit microprogram malfunctions. Some of the other additional sense bytes identify the control unit and tape unit by serial number, optional features, and engineering change (EC) level.

Two other very significant new serviceability features are microdiagnostics resident in the 3803 control unit and radial attachment of 3420 tape units to the 3803.

Resident microdiagnostics in the 3803 enhance test operations for the 3803/3420 subsystem by relieving the CPU of the execution of most time-dependent tests. Diagnostics in the 3803 are executed via use of a diagnostic command issued by a program.

The 3803 also contains diagnostics that are operative during normal tape processing operations. These diagnostics perform operations such as the monitoring of measurement functions of the tape units. If an irregularity is noted, the control unit generates sense bits to inform the executing program of the malfunction.

Tape subsystem availability is improved by radial attachment of 3420 tape units to the 3803 control unit. That is, each 3420 is cabled directly to the control unit so that any malfunctioning tape unit can be disconnected from the tape subsystem for servicing without disturbing the other tape units. When tape units are attached to the control unit in series (each tape unit cabled to the next tape unit), as are 2400-series units, the entire tape subsystem must be taken offline to uncable a tape unit.

These new features, combined with the use of fewer adjustable parts, are designed to provide optimum tape subsystem availability through better reliability and reduced maintenance time.

In conclusion, the 3803/3420 Magnetic Tape Subsystem offers Model 30 and 40 users of 2401 tape units the following advantages:

- Increased throughput for tape operations because of faster data rates, faster access times, and less rewind time for short files. In-flight correction of single-bit read errors eliminates a backspace and reread procedure and reduces the number of permanent read errors.

- Reduced tape setup time because of automatic tape threading and cartridge loading

- Reduced tape library size because of 1600-BPI density

- Less tape media wear as a result of the transport design and automatic threading and less tape damage caused by handling if wraparound cartridges are used for tape volume mounting and storage

- Reduced maintenance time because of the transport design (fewer adjustable parts), functional packaging of components, expanded sense bytes, and microdiagnostics resident in the control unit

- Increased tape subsystem availability because of reduced maintenance requirements

- Compatibility with existing 2400-series tape volumes and programs

These advantages, combined with lower subsystem cost and compact, flexible tape-switching capability, make the 3803/3420 Magnetic Tape Subsystem the natural growth path for tape users.

Table 20.25.2. 3420 and 2401 Magnetic Tape Unit characteristics

	3420 Tape Units			2401 Tape Units					
Characteristic	Model 3	Model 5	Model 7	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
Data rate (KB)	120	200	320	30	60	90	60	120	180
Density (bytes/inch)	1600	1600	1600	800	800	800	1600	1600	1600
Tape speed (inches/sec.)	75	125	200	37.5	75	112.5	37.5	75	112.5
Recording technique	PE	PE	PE	NRZI	NRZI	NRZI	PE	PE	PE
Nominal interblock gap size in inches (nine-track)	.6	.6	.6	.6	.6	.6	.6	.6	.6
Nominal read access to data (ms)	4.0	2.9	2.0	16	8	5.3	16	8	5.3
In-column rewind	Yes	Yes	Yes	No	No	No	No	No	No
Nominal rewind and unload time (secs.)	76	66	51	132	90	66	132	90	66
Nominal rewind to ready status--full 2400-foot reel (secs.)	70	60	45	180	84	60	180	84	60
Automatic threading	Standard	Standard	Standard	Not available	Not available	Not available	Not available	Not available	Not available
Time to ready status after load button pressed (secs.)	10	10	7	-	-	-	-	-	-
Cartridge loading (10.5-inch reels only)	Standard	Standard	Standard	Not available	Not available	Not available	Not available	Not available	Not available
Automatic reel latch	Yes	Yes	Yes	No	No	No	No	No	No

Table 20.25.2. 3420 and 2401 Magnetic Tape Unit characteristics (continued)

	3420 Tape Units			2401 Tape Units					
Characteristic	Model 3	Model 5	Model 7	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
Recording medium (1/2-inch magnetic tape)	IBM Series/ 500 Dynexcel, Heavy Duty, or equivalent. 10.5",8.5", 6.5" reels. (Use of Mylar* is not recommended.)	Same as Model 3	Same as Model 3	Same as 3420 plus Mylar	Same as Model 1	Same as Model 1	Same as 3420	Same as 3420	Same as 3420
Inverted tape path, single-capstan drive optical tachometers	Yes	Yes	Yes	No	No	No	No	No	No
Error checking									
Single-track corrections during reading	Automatic	Automatic	Automatic	Programmed	Programmed	Programmed	Automatic	Automatic	Automatic
Vertical redundancy check	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Longitudinal redundancy check	No	No	No	Yes	Yes	Yes	No	No	No
Number of sense bytes	24	24	24	6	6	6	6	6	6
Microdiagnostics in control unit	Yes	Yes	Yes	No	No	No	No	No	No
Separate erase head	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Data security erase command	Yes	Yes	Yes	No	No	No	No	No	No
Seven Track feature	Optional	Optional	Optional	Optional	Optional	Optional	Not available	Not available	Not available
Recording technique	NRZI	NRZI	NRZI	NRZI	NRZI	NRZI	-	-	-
Densities (BPI)	800 556 -	800 556 -	800 556 -	800 556 200	800 556 200	800 556 200	- - -	- - -	- - -

----

*Trademark of E. I. Dupont de Nemours & Co. (Inc.)

Table 20.25.2. 3420 and 2401 Magnetic Tape Unit characteristics (continued)

Characteristic	3420 Tape Units			2401 Tape Units					
	Model 3	Model 5	Model 7	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6
Data rate (KB)									
800 BPI	60	100	160	30	60	90	-	-	-
556 BPI	41.7	69.5	111.2	20.8	41.7	62.5	-	-	-
200 BPI	-	-	-	7.5	15	22.5	-	-	-
IBG size (inches)	.75	.75	.75	.75	.75	.75	-	-	-
Translator	Standard	Standard	Standard	Standard	Standard	Standard	-	-	-
Data Converter	Standard	Standard	Standard	Optional	Optional	Optional	-	-	-
Dual Density feature (800/1600 BPI)	Optional	Optional	Optional	Not available	Not available	Not available	Optional	Optional	Optional
Data rate (KB) at 800 BPI	60	100	160	-	-	-	30	60	90
Recording technique at 800 BPI	NRZI	NRZI	NRZI	-	-	-	NRZI	NRZI	NRZI
IBG size at 800 BPI (inches)	.6	.6	.6	-	-	-	.6	.6	.6
Control unit	3803 with optional Seven Track or Dual Density feature (not both). Read while write (RWW) capability is not provided	Same as Model 3	Same as Model 3	2803, 2804 (RWW) Model 1 with optional Seven-Track Compatibility feature. 2803, 2804 Model 2 with optional Seven-Track, Nine-Track, or Seven- and Nine-Track Compatibility feature.	Same as Model 1	Same as Model 1	2803, 2804 (RWW) Model 2 with optional Seven-Track, Nine-Track, or Seven- and Nine-Track Compatibility feature.	Same as Model 4	Same as Model 4
Tape switching	2 x 16 3 x 16 4 x 16 (Switching features in 3803)	Same as Model 3	Same as Model 3	2 x 16 3 x 16 4 x 16 (Requires one or two 2816 units)	Same as Model 1	Same as Model 1	Same as Model 1	Same as Model 1	Same as Model 1
Two-Channel Switch	Optional	Optional	Optional	Optional on 2803 Model 1 for Models 67 and MP65 systems only.	Same as Model 1	Same as Model 1	Not Available	Not Available	Not Available

20:27 THE 3410/3411 MAGNETIC TAPE SUBSYSTEM

The 3410/3411 Magnetic Tape Subsystem extends the new dimension of price performance and reliability, availability, and serviceability provided by the 3803/3420 Magnetic Tape Subsystem to users of half-inch tape who require data rates of less than 120 KB. The 3410/3411 subsystem offers phase-encoded data recording and data rates of 20 KB, 40 KB, and 80 KB at 1600-BPI density for Models 1, 2, and 3, respectively. In addition, the totally new, compact physical design of this subsystem minimizes floor space requirements while still permitting some flexibility in the arrangement of the units within the subsystem.

The 3410/3411 subsystem attaches to System/370 Models 135, 145, 155, and 158 and to System/360 Models 22 through 50. It offers significant advantages to Model 30 and 40 users with 2415 or 2401 Models 1, 2, and 4 in their tape configuration. The tape commands, status responses, and basic sense data of this tape subsystem are compatible with those of 2400-series and 3420 tape units. Therefore, existing Model 30 and 40 programs for 2400-series tape units will operate without change on the Model 145 (subject to the restrictions stated in Section 10:05) to handle operations on 3410/3411 subsystems with comparable features. System/360 users without any tapes installed will find that the 3410/3411 subsystem offers lower cost entry into tape processing.

The 3410/3411 subsystem consists of Models 1, 2, and 3 of the 3411 Magnetic Tape Unit and Control and Models 1, 2, and 3 of the 3410 Magnetic Tape Unit. A 3410 consists of a single tape unit so that a subsystem can include an odd number of units (a 3410 advantage when compared to 2415 configurations). One 3411 must be part of every subsystem since the 3411 contains the tape control functions and power supply required to operate its own tape unit and the 3410 tape units that can be attached to it.

Each model of the 3411 can control only one tape speed. Therefore, different models of a 3410 tape unit cannot be attached to a given 3411 model. A 3410/3411 Model 1 subsystem can consist of from one to four tape units--one 3411 Model 1 unit and up to three 3410 Model 1 units. A 3410/3411 Model 2 or Model 3 subsystem can include from one to six tape units--one 3411 unit and up to five 3410 units, all of the same model as the 3411. The Additional Tape Units feature is required for the 3411 when more than three 3410 tape units are attached. A 3410 Model 1 can be field-upgraded to a Model 2 or 3 while a Model 2 can be field-upgraded to a Model 3. Field upgrades of 3411 models are possible also.

A significant advantage of the 3410/3411 tape subsystem is the relatively smaller amount of floor space it requires when compared with 2400-series tape units. This reduction is achieved by the integration of the control unit with one of the tape units, the compact physical size of the units themselves, and a new unit design that reduces the amount of floor space required for service clearance.

The 3410 tape unit is shown in Figure 20.27.1. The 3411 and 3410 have the same dimensions: 31 inches across the front, 32 inches high in the front, 39 inches high in back, and 27 inches deep. A pair of units in a 3410/3411 subsystem (including tape control) is approximately the same width across the front as a pair of 2415 tape units with tape control (2415 Model 1 or 4) and as two 2401 tape units, which require a stand-alone control unit. The depth of a 3410 is about the same as that of a 2401 and a 2415 but the 3410 is slightly more than half the height of 2400-series units.

The 3411 and 3410 units have been designed such that normal servicing can be accomplished by access to the front of the unit. Thus, less clearance space is required, and the rear of any unit or of the subsystem itself can be located as close as six inches to a wall. No clearance is required on the right or left side of the subsystem.

Flexibility in subsystem arrangement is provided. Units in the subsystem must be physically attached to one another at the front corner. The 3411 can have 3410 units attached to either or both of its sides. Any two units can be placed side by side or can have an angle of up to 90 degrees between them. Figure 20.27.2 illustrates some of the permissible subystem layouts.

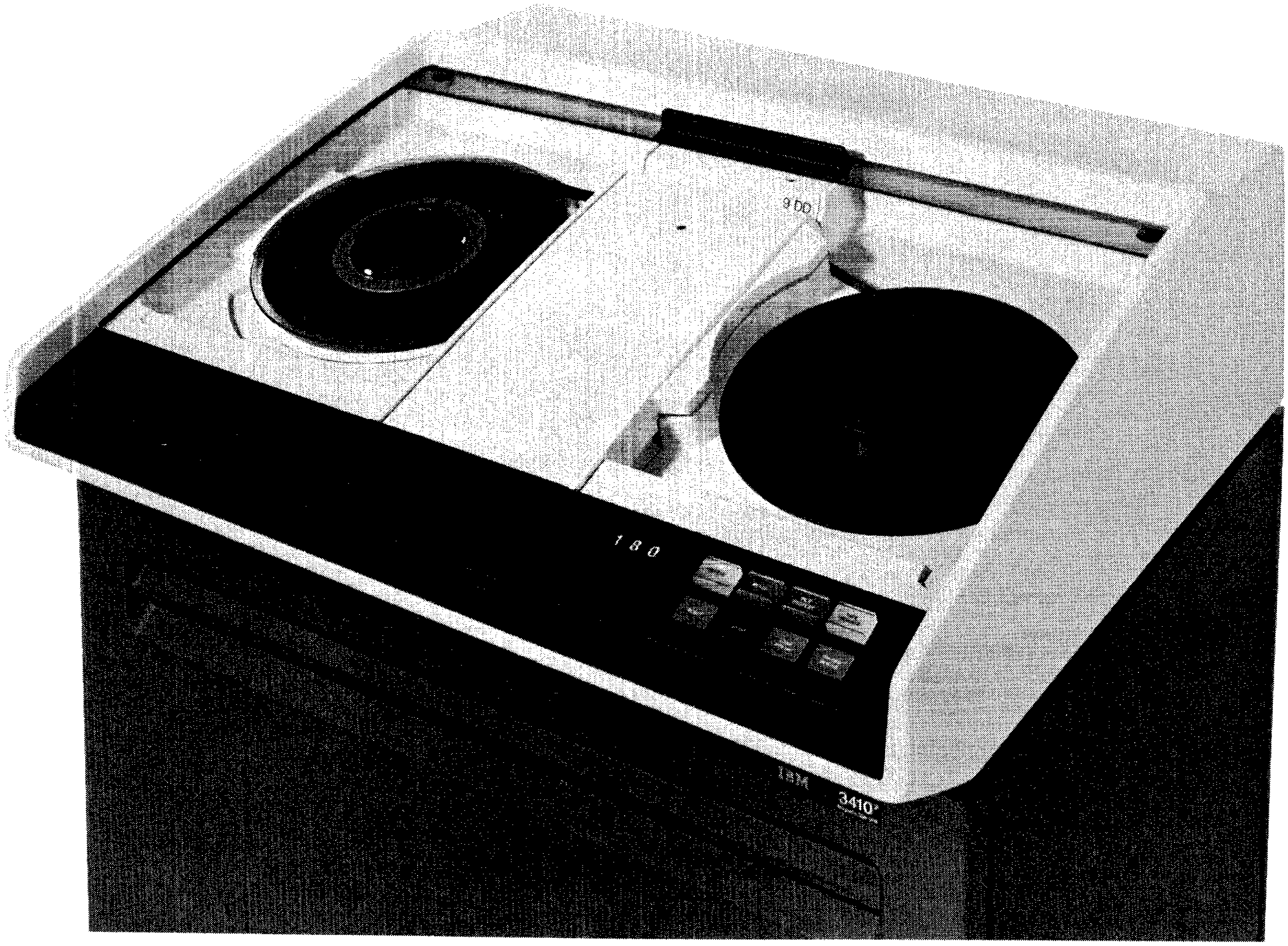


Figure 20.27.1. The 3410 Magnetic Tape Unit

As in a 3803/3420 tape subsystem, 3410 tape units are radially, rather than serially, attached to the tape control unit so that a malfunctioning tape unit can be disconnected from the subsystem for servicing, parts replacement, etc., without disturbing the other tape units. A feature unique to the 3410/3411 tape subsystem, however, is the use of internal cabling to connect 3410 units to the 3411, instead of under-the-floor cabling.

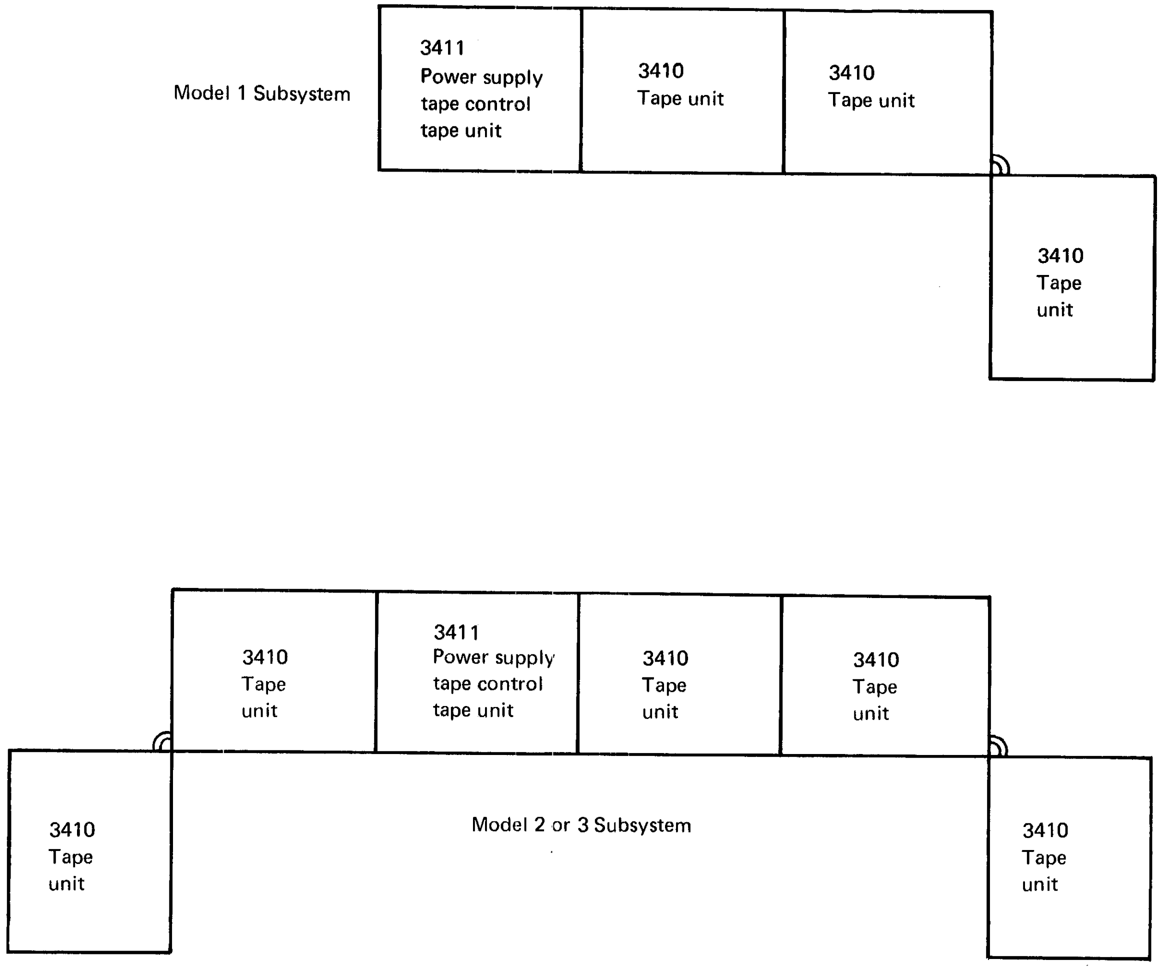


Figure 20.27.2. 3410/3411 tape subsystem physical layouts

The 3410 tape unit is designed to minimize tape mounting time and reduce wear on the tape medium. A tape reel is mounted on the left-hand side of the tape transport. A new push-pull type of quick release latch is used. As shown in Figure 20.27.1, the tape path in a 3410 has been simplified so that the operator need thread the tape only across the top of the transport, instead of under two rollers as well as the read/write head. A single-capstan drive similar to that of the 3420 is used, which minimizes tape wear. An optical tachometer provides more precise control of tape motion, as compared with previous tape units, to ensure accurate tape starts and stops and interblock gaps of correct size.

Dual Density and Seven Track optional features are available for 3410/3411 Models 1, 2, and 3. The Dual Density feature enables a drive to handle both nine-track 1600-BPI PE and nine-track 800-BPI NRZI operations. Seven-track 200/556/800-BPI NRZI operations can be handled on a drive with the Seven Track feature installed. These features are mutually exclusive and field installable. If either the Dual Density or Seven Track feature is installed in a 3410 tape unit or the tape unit portion of the 3411, the companion feature must be installed in the tape control portion of the 3411. The translate and data convert functions are included with the Seven Track feature.

A data security erase command is provided for the 3410/3411. This command is not provided for 2400-series tape drives. When a data security erase command is issued, the tape unit selected erases tape from the point at which the operation is initiated until the tape indicate (end-of-file) marker is sensed.

The 3410/3411 tape subsystem has many of the reliability, availability, and serviceability features offered by the 3803/3420 subsystem. Phase-encoded recording and radial tape unit attachment have already been mentioned. (Phase-encoded recording is explained in Section 20:25.) The 3411, like the 3803, also uses digital signals, instead of analog, to transfer data across the I/O interface, which increases subsystem reliability by reducing its sensitivity to noise caused by power fluctuations, static, etc.

The tape control portion of the 3411 is implemented in monolithic technology and is a smaller version of the 3803 control unit that offers many of the same advantages of 3803 technology. The 3411 provides more sense bytes than the 2400-series tape units (nine instead of six) and improved serviceability via microdiagnostics contained in the control unit. Tape motion microprograms can be executed by the customer engineer when the 3411 is switched to offline status.

Table 20.27.1 compares the features of 3410, 2415, and 2401 tape units.

In summary, users of 2401 Models 1, 2, and 4 and 2415 tape units will find the 3410/3411 tape subsystem offers the following advantages:

- Improved price performance

- Minimized floor space requirements

- Simplified tape threading and a push-pull hub to minimize tape setup time

- Improved RAS features

- Compatibility with 2400- and 3400-series tape units

Table 20.27.1. 3410. 2401, and 2415 Magnetic Tape Unit characteristics

	3410 Tape Units			2401 Tape Units			2415 Tape Units	
Characteristic	Model 1	Model 2	Model 3	Model 1	Model 2	Model 4	Models 1,2,3	Models 4,5,6
Data rate (KB)	20	40	80	30	60	60	15	30
Density (bytes/inch)	1600	1600	1600	800	800	1600	800	1600
Tape speed (inches/sec.)	12.5	25	50	37.5	75	37.5	18.75	18.75
Recording technique	PE	PE	PE	NRZI	NRZI	PE	NRZI	PE
Nominal interblock gap size in inches (nine-track)	.6	.6	.6	.6	.6	.6	.6	.6
Nominal gap passing time (ms)	48.0	24.0	12.0	16	8	16	32.0	32.0
Nominal read access (ms)	15.0	12.0	6.0	14.7	7.9	16.4	11.5	12.8
In-column rewind	Yes	Yes	Yes	No	No	No	Yes	Yes
Nominal rewind time 2400-foot reel (secs.)	180	180	120	132	90	132	240	240
Reel latch	Push-pull hub	Push-pull hub	Push-pull hub	Quick release latch	Quick release latch	Quick release latch	Quick release latch	Quick release latch
Recording medium (1/2-inch magnetic tape)	IBM Series/500, IBM Dynexcel, IBM Heavy Duty, or competitive formulations that meet the tape reel criteria in Tape Specifications (GA32-0006). IBM tapes other than the above do not provide adequate reliability and should not be used. 10.5", 8.5", 6.5" reels.	Same as Model 1	Same as Model 1	Same as 3410 plus Mylar*	Same as Model 1	Same as 3410	Same as 3410 plus Mylar	Same as 3410

*Trademark of E. I. Dupont de Nemours & Co. (Inc.)

Table 20.27.1. 3410, 2401, and 2415 Magnetic Tape Unit characteristics (continued)

	3410 Tape Units			2401 Tape Units			2415 Tape Units	
Characteristic	Model 1	Model 2	Model 3	Model 1	Model 2	Model 4	Models 1,2,3	Models 4,5,6
Single-capstan drive and optical tachometer	Yes	Yes	Yes	No	No	No	No	No
Error checking								
Single-track corrections during reading (PE mode)	Automatic	Automatic	Automatic	Programmed	Programmed	Automatic	Programmed	Automatic
Vertical redundancy check	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Longitudinal redundancy check	PE mode-No NRZI-Yes	PE mode-No NRZI-yes	PE mode-No NRZI-Yes	Yes	Yes	No	Yes	No
Number of sense bytes	9	9	9	6	6	6	6	6
Microdiagnostics in control unit	Yes	Yes	Yes	No	No	No	No	No
Separate erase head	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Data security erase command	Yes	Yes	Yes	No	No	No	No	No
Seven Track feature	Optional	Optional	Optional	Optional	Optional	Not available	Optional	Optional
Densities (BPI)	800 556 200	800 556 200	800 556 200	800 556 200	800 556 200	- - -	800 556 200	800 556 200
Data rate (KB)								
800 BPI	10	20	40	30	60	-	15	15
556 BPI	6.9	13.9	27.8	20.8	41.7	-	10.4	10.4
200 BPI	2.5	5.0	10.0	7.5	15	-	3.7	3.7
Recording technique	NRZI	NRZI	NRZI	NRZI	NRZI	-	NRZI	NRZI
IBG size (inches)	.75	.75	.75	.75	.75	-	.75	.75
Translator	Standard	Standard	Standard	Standard	Standard	-	Standard	Standard
Data Converter	Standard	Standard	Standard	Optional	Optional	-	Optional	Optional

Table 20.27.1. 3410, 2401, and 2415 Magnetic Tape Unit characteristics (continued)

Characteristic	3410 Tape Units			2401 Tape Units			2415 Tape Units	
	Model 1	Model 2	Model 3	Model 1	Model 2	Model 4	Models 1,2,3	Models 4,5,6
Dual Density feature (800 - 1600 BPI)	Optional	Optional	Optional	Not available	Not available	Optional	Not available	Optional
Data rate (KB) at 800 BPI	10	20	40	-	-	30	-	15
Recording technique at 800 BPI	NRZI	NRZI	NRZI	-	-	NRZI	-	NRZI
IBG size at 800 BPI (inches)	.6	.6	.6	-	-	.6	-	.6
Control unit	3411 includes one tape unit. One to three 3410 tape units can be added. Dual density is optional.	3411 includes one tape unit. One to five 3410 tape units can be added. Dual density is optional.	Same as Model 2	2803, 2804 (RWW) Model 1. Up to 8 tape units (Models 1,2, 3) can be attached. Seven-Track Compatibility feature is optional. 2803, 2804 Model 2 with optional Seven-Track, Nine-Track, or Seven- and Nine-Track Compatibility feature.	Same as Model 1	2803, 2804 (RWW) Model 2. Up to eight tape units (Models 4, 5, 6) can be attached. Seven-Track, Nine-Track, or Seven- and Nine-Track Compatibility feature is optional.	Control integrated with tape units. Model 1 has 2 units and control, Model 2 has 4 units and control, Model 3 has 6 units and control.	Control integrated with tape units. Model 4 has 2 units and control, Model 5 has 4 units and control, Model 6 has 6 units and control.
Attachment of tape units to control	Radial	Radial	Radial	Series	Series	Series	Series	Series
Tape switching	Not available	Not available	Not available	2 x 16 3 x 16 4 x 16 (Requires one or two 2816 units)	Same as Model 1	Same as Model 1	Not available	Not available
Two-Channel Switch	Not available	Not available	Not available	Optional on 2803 Model 1 for Model 67 and MP65 systems only.	Same as Model 1	Not available	Not available	Not available

20:30 THE 3505 CARD READER AND THE 3525 CARD PUNCH

These 80-column card units provide a wide range of functional capabilities and high reliability. They are designed to minimize and simplify operator intervention. New functions such as optical mark reading, read column eliminate, and card printing, new error recovery features such as automatic punch retry and feed retry, and new diagnostic procedures are provided.

The 3505 Card Reader, available in two models, provides medium- (800 cpm) and high-speed (1200 cpm) card reading, standard column binary read capability, and optional optical mark reading. The 3525 Card Punch, available in three models, offers a range of punch speeds (100, 200, or 300 cpm) as well as optional card reading and card printing. Complete configuration flexibility is provided. Any speed model of the punch, with any combination of its options, can be combined with either speed model of the reader, with any combination of its options, to provide the speeds and functions desired by an installation.

The 3505 Card Reader, shown in Figure 20.30.1, attaches to System/370 Models 135 and up and to the System/360 Model 195. The 3505 contains a microprogram-controlled control unit, which is fully buffered to prevent data overrun. The 3505 can be attached to a byte multiplexer, selector, or block multiplexer channel and can be assigned any priority. When the 3505 is attached to a byte multiplexer channel, data is transferred between the channel and the 3505 in one-byte (EBCDIC mode) or two-byte (binary mode) bursts. When the 3505 is attached to a selector or a block multiplexer channel, the entire data record is transferred in burst mode.

The 3525 Card Punch, shown in Figure 20.30.2, attaches to a channel via the 3505 Card Reader. The maximum distance between the two units is 20 feet. Only one 3525 punch can be attached via each 3505 reader, which must have a 3525 adapter installed. The 3525 adapter provides all the additional control unit hardware required for independent, fully buffered operation of the 3525. While only one control unit position on a channel is required (as for a 2540 Card Read Punch), two subchannels (on a byte or block multiplexer channel) and separate unit addresses are used. The 3505 and 3525 appear to the channel to be two logically and physically independent devices. Although the reader and the punch use the same control unit in the 3505, most failures can be diagnosed and repaired on one unit while the other continues to operate.

Table 20.30.1 lists the features of these new units. Table 20.30.3 at the end of this subsection compares the features of the 3505 reader and the 3525 punch with those of the 2540 reader/punch.

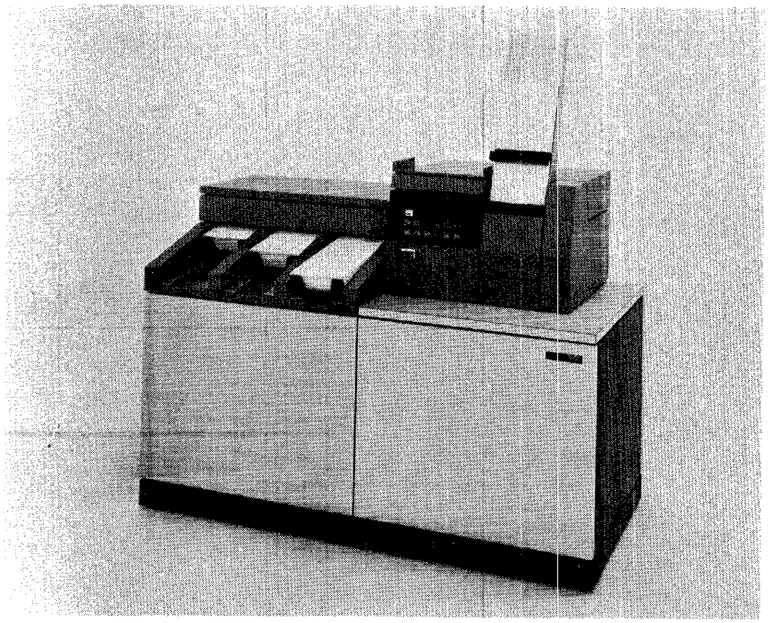


Figure 20.30.1. The 3505 Card Reader

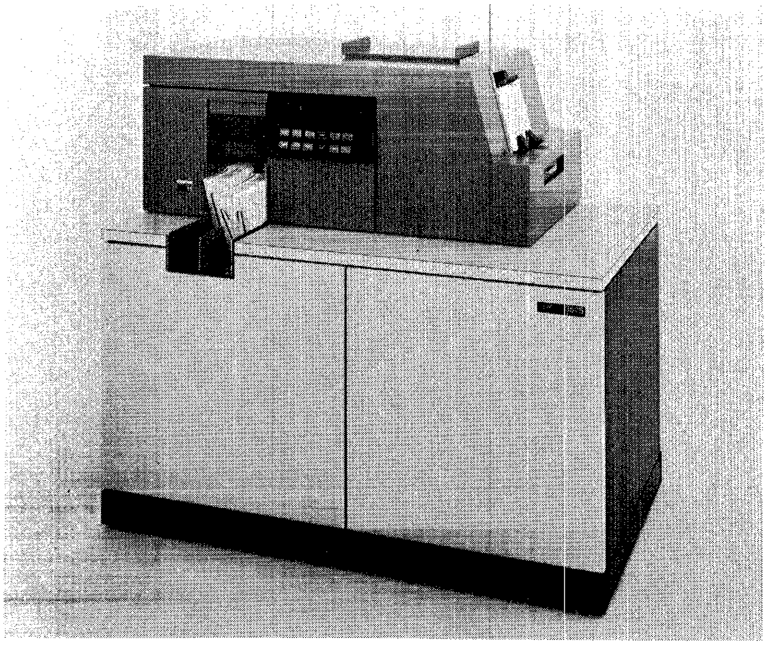


Figure 20.30.2. The 3525 Card Punch

Table 20.30.1. 3505 Card Reader and 3525 Card Punch features

Characteristic	3505 Card Reader	3525 Card Punch	3525 Card Punch with Card Read Option
Models and maximum rated speeds	B1 - 800 cpm read B2 - 1200 cpm read (Includes fully buffered control unit and attaches directly to a channel)	P1 - 100 cpm punch P2 - 200 cpm punch P3 - 300 cpm punch (Attaches to channel via 3505)	P1 - 100 cpm punch and read P2 - 200 cpm punch and read P3 - 300 cpm punch and read (Attaches to channel via 3505)
Standard features	• 3000-card-capacity file feed • One logical stacker consisting of two physical stackers with a 1750-card capacity each (alternate stacking feature) • Feed retry • Card Image • Read Column Eliminate	• 1200-card-capacity hopper • Two 1200-card-capacity stackers • Punch column binary • Automatic punch retry and dedicated error stacker	• 1200-card-capacity hopper • Two 1200-card-capacity stackers • Punch and read column binary • Automatic punch retry and dedicated error stacker (in non-read/punch mode) • Read Column Eliminate
Optional features (field installable unless otherwise noted)	• Optical Mark Reading • Selective Stacker • 3525 Punch Adapter • 3525 Read Punch Adapter • 3525 Two-Line Print Control • 3525 Multi-Line Print Control • 51/80-Column Interchangeable Read Feed on Model B2 (not recommended for field installation)	• Two-Line Card Print (not recommended for field installation) • Multi-Line Card Print - up to 25 lines (not recommended for field installation)	• Two-Line Card Print (not recommended for field installation) • Multi-Line Card Print - up to 25 lines (not recommended for field installation)

THE 3505 CARD READER

Models B1 and B2 of the 3505 differ only in card reading speed, and model changes can be made at an installation. All optional features for the 3505, except the 51/80-Column Interchangeable Read Feed, are also field installable.

The I/O commands used for card reading on the 2501 and 2540 are a compatible subset of those used on the 3505. Additional commands are required for the 3505 to handle new features, such as Read Column Eliminate and Optical Mark Reading.

The 3505 offers several new features designed to enhance card reading. First, friction feeding with vacuum-assist is used to feed cards instead of the picker knives used on the 2540 because the latter becomes less effective as card reading speeds increase. When a feed operation is performed in the 3505, it causes the bottom card in the read hopper to drop down on a continuously rotating feed roll. The combination of feed roll motion and vacuum-assist causes the card to be fed to the first station. The use of friction feeding makes the 3505 less sensitive to feeding cards with frayed edges than if picker knives were used.

Card feeding, as implemented in the 3505, also provides another advantage. Feed control is completely asynchronous; there is no waiting for clutch-points as there is for the 2540. A feed is initiated as soon as the feed command is received. Therefore, the 3505 can achieve average card reading speeds that are closer to the maximum rated speed of the model when the time between feed commands exceeds the 75-ms or 50-ms cycle required to maintain 800- or 1200-cpm reading. For example, reading speed on a 2540, which uses a multitoothed clutch for card feeding, drops from 1000 cpm to 750 cpm if a feed cycle occurs every 62 ms instead of every 60 ms. On a 3505 Model B2, reading speed averages approximately 1154 cpm, instead of 1200 cpm, if a feed cycle occurs every 52 ms instead of every 50 ms.

Second, an automatic feed retry function is implemented. If a card fails to feed on the first try, three feed retries are made by the 3505 before a misfeed indication is given. This feature reduces operator intervention. Should a card jam occur during card movement, easy access to the entire card path within the 3505 helps quick removal of affected cards.

Third, a photoelectric read head, designed to provide a high degree of read reliability, is used to read cards serially by column. Optical reading is inherently more reliable than reading via brushes because a photoelectric read head is less prone to damage and/or wear than are metal read brushes. The method used for read checking gives the 3505 an improved capability of reading card columns that are offpunched or misregistered. Sixteen time counts are taken for the reading of each card column. A column is read at four of the middle 16 time counts (5, 6, 9, and 10). The four readings are then inspected for a valid combination of data or no data impulses. Data validity checking, performed when EBCDIC data is being read, is the same as that for the 2540. (More than one punch in rows 1 through 7 constitutes an invalid EBCDIC character.)

Last, Read Column Eliminate (RCE) is a standard programmable feature that permits the 3505 to ignore card columns that contain perforations or other internal scores that would normally cause a read check. The new WRITE RCE FORMAT command must be issued before card feeding and reading occurs. This command is provided to set up the format under which card columns will be read and to establish read column eliminate format mode of reading. It transfers 80 bytes of data to the control unit, indicating which columns are not to be read. Blanks are transferred to processor storage for these columns when cards are read in format mode. FEED, READ only, and READ AND FEED commands with the new format mode bit on in the operation code must then be issued to cause read column eliminate to be effective. Reading in format mode continues until the first feed type command without the format mode bit on is issued or until unit exception status (end of file) on the 3505 is accepted by the system.

Card stacking has been designed to require operator intervention less frequently. Two 1750-card-capacity stackers that functionally provide one logical stacker with a 3500-card capacity are standard. This is achieved through implementation of an alternate stacking facility. Initially, cards are placed in the first (or right-half) stacker and a light on the operator panel is turned on to indicate that the right half is the active stacker. When the right-half stacker becomes full, the right active light is turned off, cards automatically begin stacking in the second (or left-half) stacker if it is ready, and the left active light is turned on. When the left-half stacker becomes full, stacking is automatically switched back to the right-half stacker if the operator has removed <u>all</u> the cards from the right-half stacker and set the stacker ready switch on the operator panel to indicate an empty right-half stacker condition. Otherwise, a stacker-full indication is given. Thus, more than two and one-half times the number of cards can be stacked in a 3505 as in a 2500-series card reader before operator intervention is required. If care is exercised, cards can also be removed from a stacker while cards are being placed in it, as can be done with 2500-series readers.

If the optional Selective Stacker feature is installed, cards can be directed under program control to a second logical stacker (third physical stacker) with a capacity of 1750 cards. Alternate stacking into logical stacker 1 (right and left half) occurs as already described. An unlimited amount of time is available in which to issue the stacker select command. During this time the card is positioned at a wait station. A 3505 without the Selective Stacker feature installed ignores stacker select commands and places all cards in logical stacker 1. If a stacker select command indicating pocket 3 is received by a 3505 with the Selective Stacker feature, the card is placed in stacker 2. An error indication is not given in either situation.

The standard Card Image feature for the 3505 provides the same function as the Card Image or the Column Binary feature for other readers. Read column eliminate can also be operative during column binary mode reading.

The optional Optical Mark Read (OMR) feature provides the 3505 with the ability to read up to a maximum of 40 columns of vertical marks on an 80-column card. Both vertical mark fields and punched-hole fields can be read in one pass of a card. Vertical marks can be preprinted on cards in a nonreflective ink. Alternatively, marks can be made by hand with a No. 2 pencil or equivalent, thus eliminating the necessity of using special graphic pencils, as is required for mark sensing operations on the 514 Reproducing Punch and the 519 Document Originating Machine.

As shown in Figure 20.30.3, the center of a vertical mark column is coincident with a punch column. Vertical mark fields can begin at any column and can be intermixed with punched-hole fields in any sequence, subject to the following: Any vertical mark column must always be preceded by at least one blank column (except for column 1) and followed by at least one blank column (except for column 80). OMR card design, nonreflective ink requirements, marking restraints, etc., are discussed in detail in <u>IBM 3505 Card Reader and IBM 3525 Card Punch Subsystem</u>, GA21-9124.

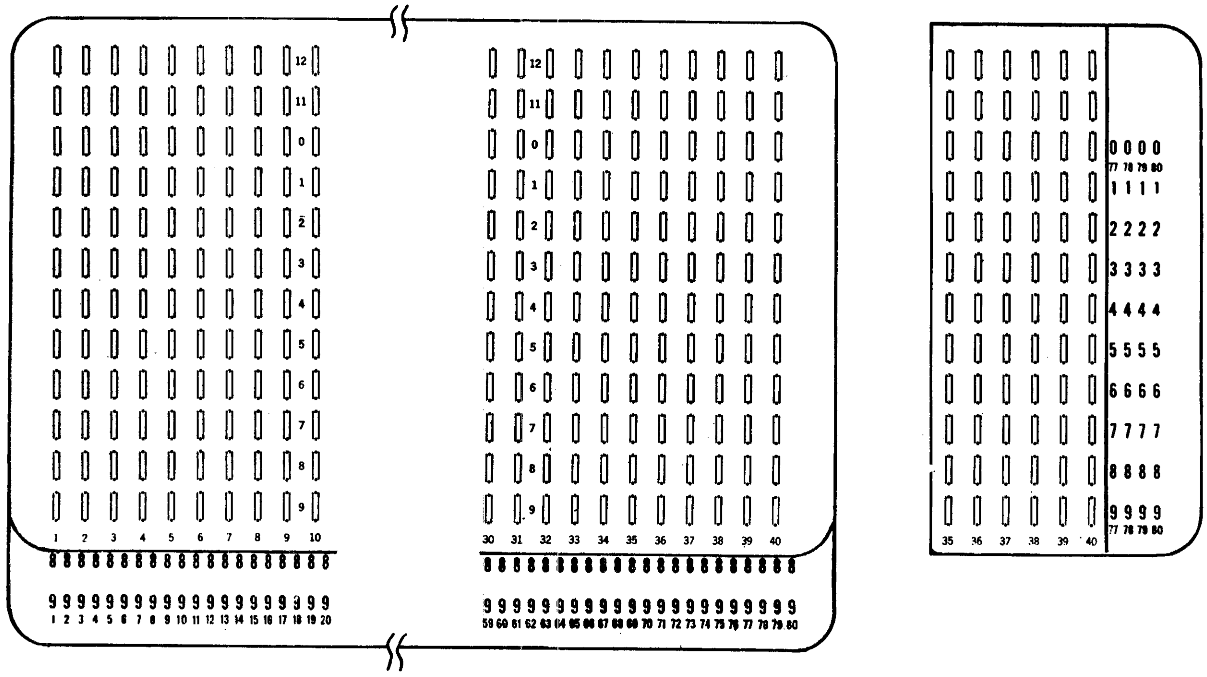


Figure 20.30.3. Vertical mark card format

OMR mode and format are established in the same manner as they are for read column eliminate. The new WRITE OMR FORMAT command must be issued to establish OMR format reading mode and to indicate which columns are to be read in OMR mode. FEED, READ only, and READ AND FEED commands must have the format mode bit on in the operation code to cause optical mark reading to be effective. OMR and read column eliminate cannot be used simultaneously (because they use the same format hardware and command mode bit).

Vertical mark fields can be read in EBCDIC or column binary mode, and validity checking is performed on EBCDIC data as usual. A vertical mark field transferred to storage does not contain the interspersed blank columns contained in the card. The control unit compresses vertical mark fields into contiguous data fields by sending only vertical mark columns to storage. Thus, a three-character vertical mark field in card columns 1 to 6 appears in positions 1 to 3 of the input area in storage.

When a vertical mark column is read, the control unit distinguishes between a marginal mark and a valid mark. A marginal mark indication can result from a poor erasure, a short mark, or a poor marking device. If a vertical mark column cannot be read properly, a hexadecimal '3F' in EBCDIC mode, or '3F3F', in column binary mode, is placed in the input area in that column position. (Hexadecimal '3F' corresponds to a 7, 8, and 9 punch in a card column. This character should not be used for any other purpose on cards to be read in OMR mode.) Column 80 (or 160) will also contain a hexadecimal '3F' when a marginal mark is sensed in any column. An error indication is not given, so that the user program can scan vertical mark fields for '3F' characters (if column 80 or 160 contains a '3F') and take the recovery action desired. If the Selective Stacker feature is installed, vertical mark cards with read errors can be selected into stacker 2, unless card sequence must be maintained.

A vertical mark field must not contain any punches or nonreflective marks in addition to the actual vertical mark data. Thus, if a card is designed to have the characters of a vertical mark field handprinted as well as vertically marked (for verification purposes, for example), the handprinted characters must not be contained in any vertical mark field. Further, any nonreflective marks made on the card should be separated from vertical mark fields by a minimum of two columns, and no writing should appear on the leading edge of the card before column 1.

OMR offers low-cost data entry and can be used in a variety of application areas. As a means of data recording, it has the following advantages:

- The only equipment required is an ordinary pencil.
- Very minimal user training is required and errors can be corrected easily by proper erasure.
- Data can be recorded at any location, quietly and without the necessity of an electrical or communications hookup.
- An unlimited number of people can record data simultaneously.

A 3505 with OMR does not provide all of the facilities of the 514 and 519, specifically, punching into marked cards and end printing, the latter as on the 519. However, the 3505 with OMR offers advantages over 514/519 mark sensing:

- Significantly greater reading speeds (up to 800 or 1200 cpm versus 100 cpm)
- Use of ordinary instead of special graphic pencils
- Up to 40 columns of data per card versus a maximum of 27
- Elimination of an offline processing operation

The availability of OMR on the 3505 provides the opportunity to (1) integrate existing 514/519 mark sensing processing into the normal stacked job operations and (2) install new applications that will benefit from the use of this method of data entry. Many of the applications for which OMR offers advantages are listed below.

Education
Test scoring
Surveys
Scheduling
Problem solving
Student registration
Attendance reporting
Library recording
Supplies requisitioning
Grade reporting

State and Local
License renewals
Assessors' reports
Traffic surveys
Statistics
Police activities
Accident reports
Public utilities
Motor vehicle inspection

Communications
Toll billing

Manufacturing
Physical inventory
Inventory receipts
Inspection
Plant maintenance reporting
Accounts receivable

Distribution
Order entry
Mail order
Physical inventory

Utilities
Meter reading

Medical
Admission data
Medical history
Menu selection
Test results
Nurses' notes

Federal
Census data
Military logistics

Airlines
Surveys
Beverage control
Inventory control

Insurance
Premium notices

Finance
Loan payment coupons

Media
Subscription fulfillment
Direct mail advertising
Book/record club promotion
Market surveys
Membership accounting

The optional 51/80-Column Interchangeable Read Feed feature enables 51-column or 80-column cards to be processed on the 3505 Card Reader Model B2. The 51 columns of data are placed in the buffer in locations 1 to 51. Other positions are set to blanks except for column 80, which is still used to indicate a marginal mark in columns 1 to 51 when the OMR feature is used. The same functions are provided for 51-column cards as for 80-column cards. Stacker capacity is limited to 1500 cards when this feature is installed, and intermixing 51- and 80-column cards in the same input operation is not supported.

THE 3525 CARD PUNCH

Either the 3525 Punch Adapter or the 3525 Read Punch Adapter must be installed on the 3505 in order to attach a model of the 3525 Card Punch to a system channel. These features can be field-installed. Models P1, P2, and P3 of the 3525 differ only in speed, and model changes can be made in an installation. Any 3525 model can be combined with either 3505 model.

The I/O commands used for the punch side of the 2540 without punch feed read capability are a compatible subset of those provided for the 3525. (The special punch feed read command defined for the 2540 is not used on the 3525, and it will cause a command reject error on the 3525.) Additional commands for the 3525 support new features such as card printing and Read Column Eliminate. Since a 3525 with the Card Read feature installed can execute the same commands as the 3505 reader (all features except optical mark reading), the 3525 can be used as a second or a backup card reader, if necessary.

In the 3525, cards move from the hopper to a parallel read station (dummy if the Card Read feature is not installed) and on to a parallel punch station at which punch checking occurs. Then cards move to a parallel print station (dummy if a card print feature is not installed), after which they are stacked into one of two standard 1200-card-capacity stackers under program control. Thus, reading, punching, and printing can occur for each card during one pass, a facility not available on other card units for System/370 models or on card units for System/360 Models 30 and up.

Card feeding on the 3525 is handled by picker knives as on a 2540 because this technique is required for parallel card reading operations when an additional aligning station is not used. Parallel rather than serial feeding is used in the 3525 to obtain the punch speeds available. As on a 3505, the entire card path in the 3525 can be exposed quickly if a card jam occurs.

A new method of punch checking, which senses punch displacements, is implemented in the 3525. Punch checking is achieved by monitoring the actual motion of each punch. The control unit compares the output from the sensors attached to each punch to the data sent to be punched and determines whether holes were actually punched and, if so, punched correctly. Each card is also checked for correct position and skew to ensure punch registration accuracy.

A significant new recovery feature of the 3525 is automatic punch retry when combined read/punch operations are not being performed. (Punch retry is effective during punch/print operations.) When a punch error is detected, the control unit directs the error card to a 200-card-capacity dedicated error stacker (stacker 3) located under the covers of the 3525 and causes the output data in the buffer to be repunched into the next card. If the retry is successful, the correctly punched card is sent to the error stacker and the stacker 3 light is turned on to indicate to the operator that cards are in the error stacker. (A correctly punched card is placed in the error stacker so that the customer engineer can compare it with the incorrectly punched card, which can be useful in identifying the punch malfunction.) Another card is then punched with the same data and stacked, after which normal operations continue. These punch retry operations are controlled entirely by the control unit. Therefore, punch retry is transparent to the channel and the program and does not require operator intervention.

If a single punch retry is unsuccessful, punching stops and an error condition is indicated. When the operator runs out the cards, the second error card is placed in stacker 1.

Because of the way in which punch retry operates, punching into prepunched or serially numbered preprinted cards should not be attempted unless the Card Read feature is present. A combined read/punch operation must also be defined so that a READ followed by a WRITE AND FEED command sequence is issued by data management. This sequence causes punch retry to be bypassed if a punch error occurs. Otherwise, when a punch retry occurred, data meant to be punched only in card N would also be punched into cards N+1 and N+2, and card N+1 would be placed in the error stacker.

The optional Card Read feature (field installable) for the 3525 provides column binary operations as a standard feature and offers a function not available with the Punch Feed Read feature for the 2540. Read column eliminate, as already described for the 3505, is standard on a 3525 with the read feature. Therefore, perforated or internally scored cards can be read from a 3525 as well as from a 3505. In addition, cards are read optically on the 3525, and thus offpunching and misregistration of cards are not as significant a factor in successful card reading as they are when read brushes are used.

Either the Two-Line Card Print or the Multi-Line Card Print optional feature can be installed on the 3525 in addition to the options already discussed. Either the 3525 Two-Line Print Control or the Multi-Line Print Control feature is required on the 3505 as well. The card print features are mutually exclusive and not recommended for field installation; however, a change from either feature to the other can be made at the installation.

Engraved type slugs, similar to those used on a 1403 Printer, are used in the print cartridge assembly in contrast to the wire matrix technique used in the 2560 Multi-Function Card Machine. The cartridge assembly provided for the 3525 contains 64 characters, and an installation can order one of two character sets, EBCDIC (63 characters including blank) or ASCII (64 characters including blank). If the ASCII cartridge is used, the program is responsible for providing ASCII encoded data in storage (that is, the control unit will not convert EBCDIC mode data from storage to ASCII mode for printing). Black and purple ink ribbons are available.

The Two-Line Card Print feature provides the 3525 with the ability to print two lines of data on a card, up to 64 characters in length, during a single pass. Card throughput for two-line printing operations equal to the rated punch speed of the 3525 model can be obtained--100, 200, or 300 cpm. The first line of print is located along the top edge of the card and the second is between punch rows 12 and 11. This corresponds to lines 1 and 3 when 25 lines are used, as shown in Figure 20.30.4.

The Multi-Line Card Print feature permits the 3525 to print up to 25 lines on a card, 64 characters in length, during a single pass. Print lines are shown in Figure 20.30.4. They are .130 inches apart, and identical in location to the 25 line locations defined for card printing using the 2560 MFCM. The lines printed on any given card are determined by programming. A print line command supplies the number of the line on which printing is to occur as well as the data to be printed and normally causes the card to be positioned in the print mechanism at the line indicated. Thus different lines can be printed on different cards. (Note that lines must be printed in ascending line number sequence and thus a given line cannot be printed on twice in one pass.) For both card print features, the data printed on each line must be supplied by the program and can include data read from a prepunched card (if the read feature is present). There is no limit to the time taken between print commands.

The card speed when printing on a given model of the 3525 with the Multi-Line Card Print feature is a function of the number of lines printed and their location. Some sample throughputs are shown below. Sixty-four characters are printed on each line.

Number of Lines Printed	Line Position	Speed in Cards per Minute		
		P1	P2	P3
1	1	100	200	300
2	1 and 3	100	200	240
3	11, 12, and 13	67	133	150
4	11, 12, 13, and 14	67	114	120
6	11 through 16	57	89	92
10	11 through 20	44	62	63
25	All	24	29	29

A technique called overlapping is used during print operations in order to increase card throughput. During printing operations on a card for all lines except the last two, the card is successively moved to the line position at which printing is to occur while other cards in the transport remain motionless. Printing of the last two, or only two, lines occurs during the next card feed cycle during which all cards in the transport move. In order to implement overlapping, the control unit stores the data for the last two lines to be printed (in buffers in the control unit) when the last two print commands are issued but does not move the card or print the lines. Thus, the time that would normally be taken to position the card and print the last two lines is saved.

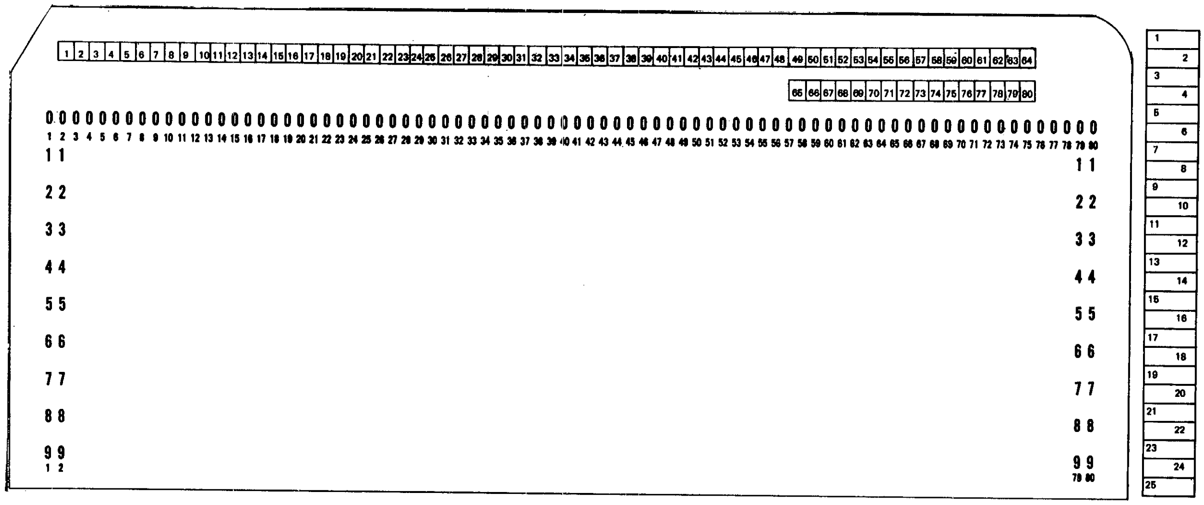


Figure 20.30.4. Card layout for 25 print lines

Following are many of the application areas in which card printing is used:

Manufacturing
Shop packet cards
Labor reporting
Material move control
Inventory issues and receipts
Physical inventory
Inspection
Plant maintenance reporting
Accounts payable
Accounts receivable

Insurance
Premium notices

Finance
Loan coupons
Christmas club

Utilities
Bills

Federal
Bonds
Notices

State and Local
Warrants
License applications
Vehicle titles
Record abstract requests

Media
Subscription fulfillment
Direct mail advertising
Book/record club promotion
Market surveys
Membership accounting

Airlines
Credit card billing
Cargo revenue accounting
Beverage control
Maintenance - shop control

Education
Test scoring
Attendance
Supplies requisitioning
Grade reporting
Class card preparation

Process
Petroleum billing

General
Proxy notices
Dividends
Program cards

Distribution
Accounts receivable

Table 20.30.2 compares the card print features offered by the 3525 Punch, the 557 Interpreter, and the 1404 Printer. Minimally, the card print capability of the 3525 can be used for card interpreting functions as a replacement for the 557. The 3525 offers increased printing throughput over the 557 because online printing on the 3525 eliminates a separate interpreting operation, multiple lines can be printed in one pass of the card, and one or two lines can be printed faster than on a 557. In addition, data not punched in the card can be printed on the 3525. The 3525 with card-printing capability can also be used as an alternative to 1404 cut-card printing operations, which are not supported by OS or DOS data management.

ERROR RECOVERY PROCEDURES FOR THE 3505 AND 3525

Error recovery procedures have been redesigned to simplify, reduce, or eliminate operator intervention. The amount of programmed error recovery required is also reduced because of the many new functions implemented in the control unit in the 3505. These include the automatic feed retry and punch retry features already discussed.

The significant new feature of error recovery for these new units is that if the control unit itself cannot correct a failure, it identifies not only the error but the specific recovery action to be taken by the error recovery procedure (ERP), the operator, or both.

When a unit error (unit check in the CSW) occurs, the control unit presents four sense bytes to the ERP (instead of one as on the 2540). The first two bytes specifically identify the error and indicate which of three possible recovery actions the ERP is to take. The second two sense bytes are provided for diagnostic purposes and are discussed under "Maintenance". The three programmed recovery actions defined are not device dependent, so that the same ERP is used for both the 3505 and the 3525.

Table 20.30.2. 3525, 557, and 1404 card-printing capabilities

Characteristic	3525 Punch with Card Print	557 Interpreter	1404 Printer
Systems to which attaches	All System/370 models and System/360 Model 195	Stand-alone	System/360 Models 25 to 50
Operations possible on one pass	Read, punch, and print	Print data read	Print only
Programming support	OS and DOS	-	Cut-card operations not supported by OS and DOS
Maximum number print lines per card	2 or 25 (depending on print feature)	25	25
Maximum number lines printed per pass of a card	2 or 25 (any number 1 to 25)	1	25 (any number 1 to 25)
Number of characters per print line	64	60	69
Line selection	Programmed	Set by operator before processing begins	Programmed
Character set size	63 (EBCDIC) 64 (ASCII)	39 (EBCDIC)	Same as for 1403 Printer
Speed (cards per minute)	Prints in parallel, one line at a time. Speed depends on number of lines printed and their location. Two 64-character lines can be printed at 100, 200, or 300 cpm with two-line feature. Up to 64 characters on lines 11 to 16 can be printed at 57, 89, or 92 cpm with multi-line feature.	100 for 1 line	Up to 800 with dual carriage

The recovery actions to be taken by the ERP (as indicated in sense byte 1) are the following:

1. Post permanent error. The ERP posts the error as permanent without attempting any recovery. (An invalid command is an example of such an error.)

2. Retry once without operator intervention (called automatic retry). The ERP reissues the failing command once and normal processing continues if the operation is successful. If one retry is unsuccessful, permanent error is posted without further recovery attempts. (Since unusual command sequences, as defined

for the 2540, are not defined for the 3505 and 3525, when an equipment check occurs, the failing command, say a read, can be reissued to attempt to correct the error. Equipment check is considered to be a permanent error on the 2540.)

3. Retry after operator intervention. When the unit is readied by the operator after the indicated recovery action has been performed, the ERP reissues the failing command. In this case, the operator could have attempted to correct the error or could have indicated that correction was not possible. If the error--a card jam, for example--was corrected, processing should continue normally when the failing command is reissued. However, if the operator determines that correction is not possible, such as when a card contains an invalid character, he can press the new permanent error key on the operator panel. When the ERP reissues the failing command after this key has been pressed, an error condition results as soon as the reissued command is received, and the appropriate sense byte is presented by the control unit to indicate that the operator pressed this key. Permanent error is then posted by the ERP without additional recovery action.

Implementation of the new permanent error key permits orderly abnormal termination of a job step to take place when certain uncorrectable failures occur. This key can also be used for operator communication with a user-written error routine that is entered after a permanent error occurs.

In order to pinpoint the specific recovery action to be taken when operator intervention is required, both the 3505 and the 3525 have a new backlighted panel as part of the operator panel. Buttons and lights on the respective operator panels are illustrated in Figure 20.30.5. The backlighted panel contains new lights, in addition to some of the same lights that are on the 2540. Whenever operator intervention is needed, the new operator call light goes on, together with one or more of the lights on the backlighted panel. The lights identify the action to be taken rather than the error that occurred. For example, the check card light, instead of a validity check light, goes on when a card should be inspected for an invalid character or offpunching. There is also a replace 1 light that goes on together with other lights for the situation in which one card, instead of two, should be placed back in the feed after a nonprocess run-out. If the operator wants or needs even more explicit directions (for example, when more than one action can be taken, as for a format mode reset condition), he can find written directions in the error recovery procedure box, readily accessible under the card joggler plate.

MAINTENANCE

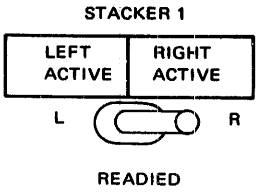
A new approach to maintenance of the 3505 and 3525 has also been taken that is designed to significantly reduce the time required to diagnose and repair failures.

When a unit error occurs on a 3505 or 3525, sense byte 2, and 3 if needed, is generated by the control unit for diagnostic purposes. Sense byte 2 is coded to uniquely identify errors. Byte 2 also serves as a pointer to the procedure to be taken as outlined in a new maintenance document, called the Graphic Integrated Manual, for customer engineers. This document contains information that currently is contained in several CE documents, and it is designed to pinpoint failures (and the repair procedure required) without the use of an oscilloscope. Resolution of the error through the use of sense byte 2 and the new manual is designed to be such that nearly all errors can be diagnosed by the CE without the use of OLTEP and OLT's.

Backlighted panel lights on 3525 Card Punch

CHIP BOX	STACKER	COVER OPEN	FEED OPEN
CHECK CARD	PRESS START	FORMAT RESET	3 CARD RUN IN
NPRO	JAM	MACHINE CHECK	PERM ERROR
OFFLINE	MISSELECT	STACKER 3	PRINT SKEW

Backlighted panel lights on 3505 Card Reader



THERMAL	STACKER	COVER OPEN	HOPPER
CHECK CARD	TRANSPORT	FORMAT RESET	REPLACE 1
NPRO	JAM	MACHINE CHECK	PERM ERROR

Buttons and lights on the Operator Panel on both the 3505 reader and the 3525 punch

PERM ERROR	OPERATOR CALL	END OF FILE	READY		POWER ON
LAMP TEST	LOG OUT	END OF FILE	START	NPRO	STOP

Figure 20.30.5. Contents of operator panels for 3505 reader and 3525 punch

The error log number contained in sense byte 2 is available to the operator. It can be displayed on the operator panel and is printed on the console by OS and DOS. If this error number is reported to IBM when repair is requested, it may enable the customer engineer to determine whether or not the failure occurred because of a broken part. If a part is broken, the customer engineer can obtain a new part before going to the installation and, thereby, considerably reduce the time the reader or punch is unavailable.

As a further diagnostic aid, the control unit for the 3505 and the 3525 contains an error log storage area for each unit into which sense byte 2, and 3 if present, is automatically stored when an error occurs. The last 14 bytes of error log data for each unit are thus maintained for the CE to use during a maintenance period. The last error logged (sense byte 2) can be displayed on the operator panel by depressing the log out key. The entire log area of a 3505 or 3525 in offline status can be displayed on the CE panel in the 3505 without disturbing other system operations. (If the 3505 is placed offline, the 3525 can still continue normal operations, and vice versa.) The log area can also be read into processor storage for analysis and printing, using an OLT.

SUMMARY

The most significant advantages of the 3505 Card Reader and the 3525 Card Punch can be summarized as follows:

- Configuration flexibility - Choice of either of two read speeds combined with any one of three punch speeds

- New functions - Optical mark reading, card printing, and read column eliminate

- New RAS features - Automatic feed retry, friction feeding, optical reading, enhanced read checking, automatic punch retry, improved error recovery procedures, and more definitive hardware failure identification provided by the control unit

- Operator-oriented design - Alternate card stacking, easier card jam removal, and explicitly defined intervention-required recovery procedures

Table 20.30.3. 3505 reader, 3525 punch, and 2540 reader/punch feature comparison

Feature	3505 Card Reader	2540 Card Read Punch (read features)	3525 Card Punch	2540 Card Read Punch (punch features)
Type of unit	Card reader and fully buffered control unit combined. Attaches to any channel. Operates independently of 3525, if the latter is attached.	Combined reader and punch unit. Requires 2821 control unit for attachment to any channel. Read and punch operations are independent and fully buffered.	Card punch unit attaches to channel via 3505. Additional control added to 3505 provides fully buffered operations independent from those of the reader.	See read features column.
Systems to which units attach	All System/370 models and System/360 Model 195	All System/370 models and System/360 Models 22 and up	Same as 3505	Same as read features column
Models and speeds	B1 - 800-cpm read B2 - 1200-cpm read (combines with any 3525 model)	1000-cpm read (one model only)	P1 - 100-cpm punch P2 - 200-cpm punch P3 - 300-cpm punch (combines with any 3505 model)	300-cpm punch (one model only)
Card reading	Optical - serial by column	Eighty read brushes parallel by row	Card Read option required. Optical reading, parallel by row.	Punch Feed Read optional feature required. Read brushes for parallel reading by row.
Card feeding	Friction feeding with vacuum-assist.	Picker knives and multi-toothed clutch	Picker knives and multitoothed clutch	Picker knives and multi-toothed clutch
Card feed retry after misfeed	Yes, up to three times	No	No	No
Read or punch checking	Each column read four times. Impulses checked for valid data or no-data indications.	Cards are read at two successive stations and hole counts are compared.	Punch checking done at punch station. Data from sensors attached to punches compared with data sent to be punched. Prepunched cards should not be used unless Card Read feature is present.	Punch checking done at 80-brush read station after punch station, as for read checking. Prepunched cards cannot be used unless Punch Feed Read feature is present.
Automatic punch retry after error	-	-	Standard hardware feature (for punch-only and punch/print operations).	Not provided by hardware. (Offered as an option by OS and DOS data management.)
File feed capacity	3000 cards	3100 cards	1200 cards	1350 cards
Stackers	One 3500-card-capacity logical stacker is standard (automatic alternate stacking in two 1750-card stackers). One additional 1750-card-capacity stacker is optional.	Two 1350-card-capacity dedicated read stackers are standard. Third 1350-card-capacity stacker is also standard and can be used by either read or punch side.	Two 1200-card-capacity stackers standard.	Two 1350-card-capacity dedicated punch stackers are standard. Third 1350-card-capacity stacker is also standard and can be used by either punch or read side.

Table 20.30.3. 3505 reader, 3525 punch, and 2540 reader/punch feature comparison (continued)

Feature	3505 Card Reader	2540 Card Read Punch (read features)	3525 Card Punch	2540 Card Read Punch (punch features)
Read column eliminate	Standard	Not available	Standard with Card Read option.	Not available
Column binary operations	Standard (Card Image feature)	Optional (Column Binary feature)	Column binary punching is standard. Column binary reading is standard when Card Read is installed.	Column binary operations are provided when the Column Binary feature is installed for the read feed.
Optical Mark Reading	Optional (up to 40 vertically marked columns per card)	Not Available	Not available (with Card Read option)	Not available (with Punch Feed Read)
51-column card read capability	Optional	Optional	-	-
Card reading on punch unit	-	-	Optional (Card Read feature) and includes column binary reading. Read Column Eliminate is standard.	Optional (Punch Feed Read feature). Read Column Eliminate is not provided.
Card printing	-	-	Optional Two-Line or Multi-Line (up to 25) Print features available	Not available
Error recovery	Control unit performs automatic feed retry. Control unit identifies error as permanent, retryable by ERP, or retryable by ERP after operator intervention. Operator informed of recovery action to be taken by lights on backlighted panel and writeup in error recovery procedures drawer. Permanent error key allows operator to terminate recovery action by ERP. Extended sense byte identifies unique errors and acts as interface to CE maintenance document.	Control unit does not perform recovery procedures. ERP inspects sense byte to determine recovery action to be taken. Lights on 2540 indicate error that occurred (no backlighted panel, permanent error key, or error recovery procedures drawer). Only one sense byte presented.	Control unit performs automatic punch retry. Other procedures are same as those for 3505. Same ERP used for reader and punch.	Same as read features column. (Punch retry is programmed and optional.)

SECTION 30: PROGRAMMING SYSTEMS SUPPORT OF THE MODEL 145 IN BC MODE

30:05 TRENDS IN DATA PROCESSING AND PROGRAMMING SYSTEMS

The Model 145 and its programming systems support have been designed to operate in the data processing environment that has been emerging since the introduction of System/360.

Significant trends are the following:

- Growth toward more multiprogramming to improve system throughput. Multiprogramming also permits the installation of new applications, such as small teleprocessing inquiry or graphics applications, that otherwise would not justify a dedicated system. Multiprogramming support also has encouraged the growth of new computer environments, as indicated by the items that follow.

- Growth of integrated emulation, that is, concurrent native and emulation mode processing on one system. The execution of emulators under operating system control improves system throughput because emulators can use control program facilities (stacked job execution, data management functions, etc.) and because native mode and emulator jobs can be scheduled to operate concurrently to utilize available system resources more efficiently. The use of integrated emulators eliminates most reprogramming and eases transition from one system to another, permitting the user to expend his efforts extending and adding applications.

- Greater use of high-level languages, such as COBOL, FORTRAN, and PL/I, for applications programming. The cost of programming has been increasing, while the cost of computing hardware has been decreasing. More productive use of programmers can be achieved by the use of high-level languages. Improvements to compile times and to the size and execution speed of code produced by high-level language translators have been made and continue to be made. The support of many more functions within high-level languages has also increased their use, and the growth of interactive computing has stimulated the addition of even more facilities.

- Growth of teleprocessing applications, such as remote inquiry, message switching, data collection, and management information systems. The ability of System/360 and System/370 to handle teleprocessing and batch processing in one system eliminates the necessity of dedicated, special purpose systems.

- Growth of remote computing activities, such as remote job entry and interactive computing (or time sharing), in both a nondedicated and a dedicated environment. Remote computing offers (1) fast turnaround for batch work submitted from remote locations, (2) remote user access to the large computing facilities and data base available at the central installation, and (3) interactive problem solving on a regular or a nonscheduled basis for personnel in locations remote from the central computer. In-house interactive computing is growing also as users attempt to use programmer time more efficiently.

- Growth toward large, online data base systems. The growth in the marketplace of remote computing, time sharing, and real-time applications necessitates the instant availability of more and more data. High-capacity, fast, low-cost, reliable direct access devices supported by appropriate data organizations, access techniques, and security measures will be required for this type of computing environment.

IBM programming systems support of these trends in data processing, by OS MFT, OS MVT, and DOS Version 4, has been growing and continues to expand. The System/370 Model 145 offers hardware, I/O devices, and performance capability required by the expanding computing environment.

30:10 OS MFT AND MVT SUPPORT

OS MFT and MVT are modified and extended as of Release 20.1 so that they support model-dependent hardware features of the Model 145 operating in BC mode. (PCP will not be extended to support model-dependent features of the Model 145.) Appropriate alteration of the resident portion of an MFT or MVT control program (nucleus) generated for a Model 145 will accommodate the BC mode fixed area of lower processor storage in the Model 145. OS MFT and MVT support of the Model 145 includes currently announced OS facilities and additional support to handle certain new Model 145 hardware features and I/O devices. Emphasis also has been placed on extending error recovery, recording, diagnostic, and repair procedures.

OS MFT and MVT support of Model 145 features and new I/O devices is provided as follows (programming systems support of RAS features is discussed in Section 50):

New instructions. Assembler F and Assembler H language translators include mnemonics for the following System/370 instructions so they can be used in user-written Assembler Language programs:

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
STORE CHANNEL ID
STORE CHARACTERS UNDER MASK
STORE CLOCK
STORE CPU ID
STORE CONTROL

An option to generate some of the new System/370 general purpose instructions is provided in ANS Full COBOL Version 3. The instructions are INSERT CHARACTERS UNDER MASK (ICM), COMPARE LOGICAL LONG (CLCL), MOVE LONG (MVCL), and SHIFT AND ROUND DECIMAL (SRP).

Extended precision floating point. Assemblers F and H, FORTRAN H, FORTRAN H-Extended, PL/I Optimizing Compiler, and PL/I Checkout Compiler language translators support extended precision floating-point.

The implementation of extended precision support by FORTRAN H and PL/I is such that the language translators and the processing programs they generate to include extended precision operations can operate on a System/370 or a System/360 with or without extended precision hardware. The language translator contains extended precision instructions and generates them for processing programs that use extended precision data. A program check interruption occurs if an extended precision instruction is executed and the feature is not present in the system. This interruption causes the processing program to call in a subroutine (the extended precision floating-point simulator) to handle extended precision operations. (The extended precision simulator, which simulates all the operations provided by extended precision instructions and extended precision divide as well, is automatically included in all operating systems at system generation time.) Extended precision divide is always simulated, since the extended precision hardware feature does not include such an instruction.

Monitoring feature. This feature is supported by the Generalized Trace Facility (GTF). GTF is a standard feature of MFT and MVT for System/370 and System/360 models. GTF uses the monitoring feature when the feature is present and simulates its function when monitoring hardware is not available. An Assembler Language mnemonic is also provided for the MONITOR CALL instruction.

GTF offers a generalized trace function, which supports more options than the existing OS trace facility, and a new trace edit function. When trace data is placed in a trace output data set instead of being maintained in processor storage, the trace edit function, a feature of IMDPRDMP which is invoked via job control and operates as a processing program, can be used to format and print the trace output. Options include printing of selected event classes, selected events within event classes, events associated only with a specific job, and events that occurred after a specific time. The trace edit function requires a minimum of 44K in which to execute.

The generalized trace function is invoked as a system task by the operator via a START command. The trace function continues until terminated by a STOP command. When the generalized trace function is active, operation of the current OS trace facility is suspended, if it is present.

The generalized trace function supports tracing of all or any combination of the following:

- All I/O interruptions, all non-PCI interruptions, or only I/O interruptions from specific devices

- All START I/O operations or only START I/O operations for specific devices

- All SVC interruptions or only interruptions for specific SVC codes

- All program interruptions or only interruptions for specific program interruption codes

- All external interruptions

- All entries into the dispatcher routine of the control program (for tracing the flow of routine execution)

- All user-supplied trace entries (discussed below)

- Events associated with the trace task

Events that occur in user-written routines can be monitored by use of the new GTRACE macro, which will expand into a MONITOR CALL instruction when used on the Model 145. Execution of this macro causes a user-supplied trace entry, containing the data presented via the macro, to be included in the trace output.

Interval timer. The interval timer is supported for the same functions as it is for System/360 except for time of day values.

Time of day clock. This clock is supported for time of day requests made by system and user tasks via the TIME macro. At IPL, the operator can validate the clock time and correct an invalid value.

Clock comparator and CPU timer feature. This feature is not supported.

Byte-oriented operands. The programmer has the ability to byte-align binary and floating-point data in Assembler Language programs, in ANS COBOL programs (by omitting the SYNCHRONIZED clause), in PL/I programs (by specifying the UNALIGNED attribute), and in FORTRAN programs. However, OS still expects parameters passed to it to be properly aligned and the high-level language translators still align unaligned fixed- and floating-point data before it is used.

1401/1440/1460 and 1401/40/60, 1410/7010 Compatibility features. Two integrated emulator programs are provided, one to emulate 1401, 1440, and 1460 programs and the other to emulate 1410 and 7010 programs. (Emulator programs are discussed in Section 40.)

OS/DOS Compatibility feature. An OS DOS emulator program is provided to support emulation of a DOS Version 3 or 4 system. (This emulator is discussed in Section 40.)

Console devices. The 3210 Model 1 and 3215 Model 1 Console Printer-Keyboards are supported as the primary operating system console device. A remote 3210 Model 2 is supported as an alternate or an additional console. (The MCS option is required to support the 3210 Model 2 as an additional console.)

Channels. The byte multiplexer channel with up to 256 subchannels is supported. Selector and block multiplexer mode are supported also. During IPL, channel mode for all installed block multiplexer channels is established via a control register channel mode bit setting based on system generation channel definitions. (The channel mode bit is discussed in Section 10:20.) The operator does not have the option of changing this mode at IPL, nor does the control program change the mode setting during system operation. The IFA and 2319 disk storage do not require any special programming and are supported in the same manner as 2314 disk storage attached to a channel.

3330-series disk storage. Strings of 3330-series drives attached via 3830 Storage Control and/or Integrated Storage Control are supported for most of the same functions as is the 2314 facility and by ASP and HASP II. The error recovery routine provided for the 3330-series includes support of the new hardware error correction features. Sixteen-drive addressing is supported to accommodate the attachment of two strings of drives to 3830 Storage Control Model 2 or to Integrated Storage Control.

RPS is supported as follows.

- The stand-alone seek issued within the I/O supervisor (IOS) is eliminated for RPS devices (3330-series drives). IOS will continue to issue stand-alone seeks for direct access devices without RPS. IOS also is capable of recognizing the channel available interruption.

- Access method support of RPS commands (SET SECTOR and READ SECTOR) is provided by QSAM, BSAM, BPAM, BDAM, ISAM, and TCAM. The sector number is calculated for fixed length records when possible or obtained by use of the READ SECTOR command (for example, during sequential processing of variable length records). Specifically, RPS is supported by:

QSAM and BSAM for all record formats and functions provided for the 3330-series except the undefined track overflow record format or if any following direct access concatenations are non-RPS devices

BPAM for processing directory and member records wherever possible unless a non-RPS device is concatenated to the PDS (directory entries are not modified to include sector values)

BDAM for direct retrieval and update of fixed-length standard and VBS format records without key, and for write verification of all BDAM record formats.

ISAM for:

1. All operations involved in data set creation (fixed- and variable-length records) using QISAM load mode (index entries will not be modified to include sector values)
2. Sequential retrieval of fixed- and variable-length prime and overflow records using QISAM scan mode (all prime records except the first on the track)
3. Addition of new records to the prime and overflow areas, including the searching of overflow chains, using BISAM
4. All validity checking operations (data and index records)
5. All updating operations (data and index records)
6. Operations that require orientation to the beginning of the track before processing, such as index searching, BISAM direct retrievals, reading the first data record during sequential operations, etc. (a sector value of zero is used)

TCAM for disk message queue processing

- Any system utility, data set utility, or IBM-supplied processing program (such as a language translator) that uses the sequential access methods supports RPS. In addition, IEHMOVE, IEBCOPY, and the initialize/dump/restore utility (IEHDASDR) include RPS support.

- The Sort/Merge (program product) supports RPS for 3330-series intermediate work devices.

- Where appropriate, RPS commands for access to SYSRES data sets is supported by:

Data set catalog routines
Direct access device space management (DADSM routines) for DSCB processing
STOW, BLDL, and FIND processing of program library (PDS) directory entries
OPEN/CLOSE/EOV processing of JFCB's in the job queue
Routines that access the job queue

Note that RPS command support is not provided for:

Program fetch
Access to TSO swap data sets
The stand-alone disk initialization and alternate track assignment routines (DASDI and IEHATLAS)

<u>The 2305 facility</u>. The 2305 facility is supported as an I/O device for most of the same functions as is 2303 Drum Storage. Rotational position sensing for 2305 facilities is supported as discussed for the 3330-series, except that an arm positioning seek is not required on 2305 modules and the 2305 is not supported as an intermediate work unit by the Sort/Merge programs. In addition, the 2305 is not supported by ASP

or HASP II or for TCAM message queues. Multiple requesting is handled by the I/O supervisor, which initiates up to seven channel programs on one 2305 module at a time. (The eighth logical device address is reserved for control purposes.) Multiple I/O operations to the same data set will be initiated if the data set is not being sequentially processed. (One operation must complete successfully before the next can be initiated during sequential processing.)

Note that specification of exchange buffering for a QSAM data set on either the 3330-series or the 2305 results in a default to simple buffering.

3211 Printer with tapeless carriage. The 3211 Printer, with or without the 18 additional print positions, is supported by QSAM and BSAM for exactly the same functions as is the 1403 Printer, and by ASP and HASP II. In addition, the control program handles loading of the Forms Control Buffer (FCB) with carriage control images. This support is similar to that provided for Universal Character Buffer (UCB) loading.

The user can define one or more default FCB images at system generation time. Two IBM-supplied default images are included automatically. All other FCB images to be used must be defined by the user and placed in SYS1.IMAGELIB, as is the case with UCS images. User-supplied default images must be identified as defaults. The FCB image to be used by a processing program can be specified in the 3211 Printer DD statement included for the job step and will be loaded into the FCB by the control program.

The FCB image currently loaded can also be changed by the programmer during execution of the processing program by use of an Assembler Language macro. If the DD statement does not specify an FCB image and the image currently loaded is not one of the defaults specified at system generation, the operator is requested to specify the FCB image to be used.

The FCB image (and the UCS image) to be loaded for a 3211 Printer used by an output writer can be indicated in the output writer procedure or in the START output writer command issued by the operator. FCB and UCB images can also be specified in the SYSOUT DD statement for a data set that is to be printed by the output writer, and they will be loaded into the 3811 control unit prior to the printing of the data set.

Any time the FCB parameter is used, as described above, the user can specify that operator verification of forms alignment is to be requested by the control program via a console message when the buffer is loaded. The operator must respond to this message.

The 3211 error recovery routine will retry a print operation after a parity check occurs in the UCB or print line buffer, if QSAM is used and three I/O buffers are provided for the printer data set. When the operation is retried, the 3811 control unit ensures that only the print positions that did not print correctly the first time are reprinted.

ASCII-mode tapes. The capability of processing ASCII-mode tapes on the Model 145, using OS, is provided by the following program products:

- ANS Full COBOL Compiler V3
- PL/I Optimizing Compiler
- PL/I Checkout Compiler
- PL/I Transient Library
- Sort/Merge
- FORTRAN IV Library (Mod I and Mod II)

- Data Set Utilities - a set of four ASCII utilities that includes printing, punching, and comparison of ASCII-mode tapes in addition to translation to and from ASCII mode
- TSO Data Utilities

ASCII-mode tapes are also supported by data management (QSAM and BSAM) so they can be processed by user-written Assembler Language programs.

The 3803/3420 Magnetic Tape Subsystem. This tape subsystem is supported as an I/O device for the same functions as 2400-series tape units. This includes support of tape switching, Seven Track, and Dual Density features. (Note that 200-BPI density tapes are not supported because the Seven Track feature includes only 556- and 800-BPI densities.) The Two-Channel Switch is supported for alternate path switching between two channels in the same system.

The 3410/3411 Magnetic Tape Subsystem. Support of this tape subsystem functionally identical to that provided for 2400-series tape units is provided.

The 3505 Card Reader. This reader is supported by QSAM and BSAM for the same basic input functions as are provided for the 2540 and 2501 readers, including support as a SYSIN device. Features are supported by the access methods as follows:

- Stacker selection into stackers 1 and 2 is provided, as for current readers. If a stacker select request for stacker 3 is received, the 3505 will place the card in stacker 2 without giving any indication of this action to the user program. If a request for stacker 2 is received and the optional Selective Stacker feature is not installed, the 3505 will direct the card to stacker 1 without any indication to the program.

- Card Image support is identical to that provided for other card readers. Either EBCDIC or column binary read mode is established at OPEN, depending on the user specification, and this mode remains in effect until the data set is closed or end of file occurs on the reader. Thus input decks that contain cards with EBCDIC and column binary punching in the same card, or a mixture of cards punched in one mode or the other, must be handled by the user with the EXCP macro.

- OMR and RCE are supported but not concurrently for the same data set. These options are requested via the DCB MODE parameter, either in the program DCB or in the DD statement DCB parameter. The format to be used is supplied by the user in the first card of the input deck. Once the data set is opened and the format has been established, the format remains in effect until the data set is closed. A data set to be read in OMR or RCE format mode cannot be placed in the input stream (in the SYSIN device). It must be read directly from the 3505 by the user program. A program that is to use one of these features must specifically request allocation of a 3505 with the feature installed, via the UNIT parameter, as the scheduler will not allocate a 3505 by feature.

The OS high-level languages (COBOL, PL/I, FORTRAN, and RPG) also support the 3505, including OMR and RCE. Support for these new features can be requested in job control statements and is, therefore, transparent to the high-level languages. The Card Image and Stacker Select features of the 3505 are supported by these languages to the same degree as they are for other readers. Any OS utility that uses QSAM or BSAM also supports the 3505 and its features. ASP and HASP II support the 3505 for the same functions they provide for other card readers.

The 3525 Card Punch. This punch is supported by QSAM and BSAM for the same basic punch and stacker selection functions that are provided for the 2540 punch. It is also supported as a SYSOUT device and, if the read feature is present, as a SYSIN device. (RCE cannot be used in the latter situation.) All optional 3525 features are also supported by the Assembler Language and by all OS high-level languages except ALGOL. The 3525 can be used to perform the following functions:

- Punching only - Either EBCDIC or binary data can be punched, and stacker selection is supported.

- Reading only - Includes support of Card Image, RCE, and stacker selection as described for the 3505 reader

- Printing only - Programming is similar to that for a printer. Functionally, a card is considered to be a page with 25 possible print lines. Varying the stacker selected by programming is not supported; however, all cards can be directed either to stacker 1 or to stacker 2.

- Interpreting only - This is actually a punch and print combination in which the data punched is also printed on line 1 (first 64 characters) and line 3 (remaining 16 characters), as shown previously in Figure 20.30.4.

- Any combination of reading, punching, and printing - The concept of associated data sets is used for handling two or more operations on the same card. A separate data set is defined by the user for each function (read, print, or punch) to be performed. These data sets are allocated to the same 3525 via the DD job control statement (AFF parameter), and when macros are issued, a certain sequence must be maintained. The advantage of this approach is that it is transparent to all high-level languages and it provides device independence. For example, a program written in a device-independent manner designed to read, punch, and print using the 3525 can be executed using a tape unit, a direct access device, and a printer for the three respective operations, if necessary, merely by changing job control statements.

OS utilities that use QSAM or BSAM support the 3525 and its optional features as described above. ASP and HASP II support the 3525 for the same functions they provide for other card punches.

30:15 DOS VERSION 4 SUPPORT

DOS support of model-dependent features of a Model 145 operating in BC mode is provided in DOS Version 3 (Releases 25 Extended and 26) and in DOS Version 4 (Release 27). Support of new Model 145 hardware features and I/O devices as of DOS Version 4 is discussed below. Features available in Version 4 but not in Version 3 are identified. DOS Version 4 support of RAS features is discussed in Section 50.

New instructions. Assembler D (14K variant) includes mnemonics for the following new System/370 instructions so they can be used in user-written Assembler Language programs:

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
STORE CHANNEL ID
STORE CHARACTERS UNDER MASK
STORE CLOCK
STORE CPU ID
STORE CONTROL

An option to generate some of the new System/370 general purpose instructions is provided in ANS Full COBOL Version 3 and ANS Subset COBOL. The instructions are INSERT CHARACTERS UNDER MASK (ICM), COMPARE LOGICAL LONG (CLCL), MOVE LONG (MVCL), and SHIFT AND ROUND DECIMAL (SRP).

Extended precision floating point. Mnemonics for extended precision instructions and data formats are added to Assembler D (14K variant). The DOS high-level language translators currently available do not support extended precision. A simulator for extended precision operations, such as that provided by OS, is not provided in DOS.

Monitoring feature. This feature is not supported except for the inclusion of a MONITOR CALL instruction mnemonic in Assembler D.

Time of day clock (Version 4 only). Support of the time of day clock for time of day values is a system generation option. When the option is included, the operator can set or correct the time of day clock during IPL. Any system or user task can then obtain the current time in microseconds by issuing the GETIME macro, which will access the time of day clock instead of the interval timer at location 80.

The time of day clock is not used by the job accounting interface; hence, when both time of day clock and job accounting support are included in a supervisor, interval timer support is included also to provide the times required by the job accounting interface. Existing user-written programs that issue the GETIME macro do not require modification in order to be used with a supervisor that has time of day clock support included.

Interval timer at location 80. The timer is used by the job accounting interface and to support time intervals as in DOS Version 3. Unless the option to support the time of day clock is included, the interval timer is also used for time of day values.

Clock comparator and CPU timer feature. This timing feature is not supported.

Byte-oriented operands. The programmer has the ability to byte-align binary and floating-point data in Assembler Language programs, in ANS COBOL programs (by omitting the SYNCHRONIZED clause), and in PL/I programs (by specifying the UNALIGNED attribute). However, COBOL and PL/I still align unaligned fixed- and floating-point data prior to its use.

1401/1440/1460 and 1401/40/60, 1410/7010 Compatibility features. Two integrated emulator programs are provided: one to emulate 1401/1440/1460 programs and the other to emulate 1410/7010 programs. (See Section 40 for a complete discussion of these emulator programs.) Emulator support of the 3330-series is provided in Version 4 only.

Consoles. The 3210 Model 1 and the 3215 Model 1 Console Printer-Keyboards are supported as the DOS console device. A remote 3210 Model 2 console is not supported.

Channels. The byte multiplexer channel and selector channel mode are supported. Block multiplexer mode is not supported. The IFA and 2319 direct access storage are supported in the same manner as 2314 disk storage attached to a channel.

3330-series disk storage (Version 4 only). Strings of 3330-series drives attached via 3830 Storage Control and/or Integrated Storage Control are supported for the same system control and system service functions as is 2314 disk storage, including support by DOS POWER II. That is, the 3330-series is supported as a system residence, system input and system output device, and by the linkage editor, librarian, and System Utility Programs (370N-UT-491). The DOS Sort/Merge program product (5743-SM1) includes support of the 3330-series as an input, output, and work unit.

Data management (SAM, ISAM, DAM, QTAM) support of the 3330-series as an I/O device, functionally identical to that available for the 2314, is provided by Assembler D (14K variant). ANS Full COBOL Version 3 Release 2, ANS Subset COBOL Release 2, PL/I Optimizing Compiler, FORTRAN IV Library Option 1, and RPG II also support the 3330-series for functions similar to those provided for the 2314.

The rotational position sensing facility is not supported. That is, SET SECTOR and READ SECTOR commands are not included in 3330-series channel programs. Block multiplexing is not supported, and a channel with 3330-series drives connected operates in selector mode. Therefore, only one data transfer channel program can be in operation on the channel at a time. However, stand-alone seek operations can be overlapped with one another and with one data transfer operation, as for a 2314, when the seek overlap option is present in the DOS supervisor.

Sixteen-drive addressing is supported to accommodate the attachment of two strings of drives to Model 2 of 3830 Storage Control or to Integrated Storage Control.

The 2305 facility. The 2305 is not supported.

3211 Printer. This printer is supported for the same functions as the 1403 Printer. POWER II support of the 3211 is provided also. New parameters are added to some of the existing Assembler Language macros to support new features of the 3211. The 18 additional print positions are supported by the Assembler Language, ANS Full COBOL Version 3, and ANS Subset COBOL. Forms Control Buffer and Universal Character Buffer loading for the 3211 are handled in the same way. The DOS operating system includes a standard image for the UCB and one for the FCB, which are loaded at IPL. If a job step requires a different image, the user must execute an IBM-supplied buffer load utility program (SYSBUFLD) as a previous job step in order to load the FCB and/or the UCB. No provision has been made for loading the FCB or UCB during execution of a job step. User-defined UCS images must be loaded from a core image library. FCB images can be loaded from cards or a core image library.

If a command retry indication is present, the 3211 error recovery routine supports retry of an operation that failed. This option must be requested by the user in the DTF for the 3211 Printer.

ASCII-mode tapes. The capability of processing ASCII-mode tapes on the Model 145, using DOS, is provided by the following program products:

- ANS Full COBOL Compiler V3 and Object-Time Library
- ANS Subset COBOL Compiler and Library
- PL/I Optimizing Compiler
- PL/I Transient and Resident Libraries
- FORTRAN IV Library Option 1
- ASCII Magnetic Tape Utilities
- Tape and Disk Sort/Merge
- RPG II

An initialize tape utility is provided that can be used to write an ASCII (or EBCDIC) mode standard volume label on a tape. ASCII-mode tapes are also supported by data management (the sequential access methods) and thus they can be processed by user-written Assembler Language programs.

The 3803/3420 Magnetic Tape Subsystem (Version 4 only). This tape subsystem is supported as an I/O device for the same functions as 2400-series tape units. This includes support of tape switching (two control units only), Seven Track, and Dual Density features. (Note that 200-BPI density tapes are not supported because the Seven Track feature includes only 556- and 800-BPI densities.) The Two-Channel Switch is supported for alternate path switching between two channels in the same system. The data security erase command is supported by the MTC command for magnetic tapes. When an MTC command with a DSE parameter is issued, tape is erased up to the location of the tape indicate marker. If the command is issued when the tape is at load point, the entire contents of the tape, including all labels, are erased up to the end-of-file marker. Such a tape must be reinitialized or have a tapemark written on it before it is used again. Utility support is provided only by System Utility Programs (370N-UT-491).

The 3410/3411 Magnetic Tape Subsystem (Version 4 only). Support of this tape subsystem functionally identical to that provided for 2400-series tape units is provided by data management. The data security erase command is supported by the MTC command for magnetic tapes, as described for the 3803/3420 Magnetic Tape Subsystem.

The 3505 Card Reader (Version 4 only). This reader is supported at the GET/PUT macro level by the Assembler Language for basic read functions, stacker selection, and column binary mode. The 3505 is also supported by all DOS high-level languages to the same extent that support for card readers is provided currently. In addition, the OMR feature is supported by ANS Full COBOL Version 3 Release 2, ANS Subset COBOL Release 2, RPG II, and the Assembler Language. Support of the RCE feature is provided in the Assembler Language and RPG II.

The 3505 is also supported as a SYSRDR, SYSIPT, or SYSIN device and by POWER II for basic read functions. If OMR or RCE is to be used by an Assembler Language program, the 3505 cannot also be used as the SYSRDR, SYSIPT, or SYSIN device.

Support of 3505 features by the Assembler Langauge is as follows:

- Stacker selection into stackers 1 and 2 is provided. If a request for stacker 3 is received, the card is placed in stacker 2 by the 3505. If a request for stacker 2 is received and the Selective Stacker feature is not installed, the 3505 places the card in stacker 1. No indication of either action is given to the user program.

- Card Image support is provided as follows. Either EBCDIC or column binary read mode is established at OPEN time, depending on the user specification, and this mode remains in effect until the file is closed. Thus input decks that contain cards with EBCDIC and column binary punching in the same card or a mixture of cards punched in one mode or the other must be handled by the user with the EXCP macro.

- Either OMR or RCE support can be requested only in the DTFCD macro. DTFDI can be used to support the 3505 only for basic read functions. The OMR or RCE format to be used must be supplied in the first card of the input deck. Once the file has been opened and the format has been established, the format remains in effect until the file is closed or until unit exception (end of file) on the 3505 is accepted by the system.

Utility support of the 3505 and 3525 functionally equivalent to 2540 support is provided by the copy disk-to-card and restore card-to-disk programs contained in System Utility Programs (370N-UT-491).

The 3525 Card Punch (Version 4 only). This punch and all its special features are supported at the GET/PUT level by the Assembler Language. The 3525 is also supported by POWER for basic punch functions. The 3525 is supported as a SYSPCH unit and, if the read feature is present, as a SYSIN device. RPG II, ANS Full COBOL Version 3, and ANS Subset COBOL Release 2 support the card print features and two or more combined operations (reading, punching, printing) on each card. RPG II also supports RCE.

The 3525 is supported by the Assembler Language and RPG II for the following functions:

- Punching only - Stacker selection is supported.

- Reading only - Includes support of Card Image, RCE, and stacker selection as described for the 3505. (RPG II support of Card Image requires inclusion of a user routine.)

- Printing only - Programming is similar to that for a printer. Functionally, a card is considered to be a page with 25 possible print lines. Varying the stacker selected by programming is not supported. All cards will be directed to stacker 1.

- Interpreting only - This is actually a punch and print combination in which the data punched is also printed on line 1 (first 64 characters) and line 3 (remaining 16 characters), as shown previously in Figure 20.30.4.

- Any combination of reading, punching, and printing - The concept of associated files is used for handling two or more operations on the same card, instead of the combined file technique currently supported for read/punch operations on a 2540 with Punch Feed Read. A separate file (read, punch, or print) is defined by the user for each function to be performed. These files are allocated to the same 3525 via ASSGN statements and their DTF's contain a new parameter to indicate that they are associated files. When macros are issued, a certain sequence must be maintained. Two or more combined operations are also supported by ANS Full COBOL Version 3 Release 2 and ANS Subset COBOL Release 2.

Utility support for the 3525 is described above under "The 3505 Card Reader".

DOS Version 3 and Version 4 users should consider moving to DOS/VS. This version of DOS supports many new functions not provided in DOS Version 4. Some of the significant new DOS/VS items are:

- A virtual storage environment (using dynamic address translation hardware)

- Up to five, instead of three, problem program partitions. The operator also has the capability of varying partition priority during system operation.

- A relocating loader that supports program phase relocation at load time

- Extensions to POWER to support certain new I/O devices and the scheduling of up to four partitions. POWER will be a component of DOS/VS and need not be ordered separately.

- Capability of using the interval timing facility in each partition in a multiprogramming environment, instead of only in one partition

- A cataloged procedures facility which allows job control to be stored in a procedure library so that it need not be placed in the input stream

- Changes in file label processing that support the interchange of tape volumes between DOS and OS

- A new data organization and access method called virtual storage access method (VSAM) that is designed to offer more function and better performance for files with additions than ISAM

SECTION 40: EMULATORS

40:05 OS 1410/7010 AND 1401/1440/1460 EMULATOR PROGRAMS

FEATURES COMMON TO BOTH EMULATORS

One of the significant new features of the Model 145 for OS users is support of integrated 1400/7010-series emulation. Only stand-alone 1400/7010 emulation is available to Model 30 and 40 OS users. A 1410/7010 Emulator program and a 1401/1440/1460 Emulator program that operate under OS control on the Model 145 are provided. These emulators operate under OS MFT, MVT, VS1, and VS2.

An emulator program requires the presence of either the IBM 1401/1440/1460 Compatibility or the 1401/40/60, 1410/7010 Compatibility feature. These no-charge options are the same features used by the two integrated DOS 1400/7010 emulator programs for the Model 145. The latter feature permits operation of both the 1401/1440/1460 emulator and the 1410/7010 emulator.

The OS 1401/1440/1460 emulator for the Model 145 provides the same functions as the OS 1401/1440/1460 emulators for System/370 Models 135 and 155. This is also true of the 1410/7010 emulators for Models 145 and 155. The 1401/1440/1460 compatibility features on Models 145 and 135 are identical but the 1400/7010 compatibility features on Models 145 and 155 differ slightly in their implementation (for SAR and SBR instructions). Therefore, a 1401/1440/1460 emulator program generated for the Model 145 will run on a Model 135 but not on a Model 155, nor will a 1410/7010 emulator generated for the Model 145 run on a Model 155. However a 1400/7010 emulator program generated for the Model 155 will run on Models 135 (1401/1440/1460 only) and 145.

The emulators operate as processing programs in MFT and MVT environments and are alike in their functional design. Those features common to both are discussed first.

The 1400/7010 emulator programs supplied for the Model 145 are relocatable and thus can operate in one or more MFT partitions or MVT regions. Any number of emulator jobs of the same or different types (1401, 1440, 1460, 1410, and 7010) can execute concurrently with System/370 jobs in the same Model 145, subject to the availability of system resources. Emulator and System/370 jobs can be intermixed in the input stream, since emulator job scheduling, initiation, and resource allocation are handled by OS job management routines. I/O operations are handled by OS data management. Emulator jobs are executed by job priority as is any OS job.

Integrated emulation provides a number of advantages over stand-alone emulation that can increase system throughput. The ability to execute 1400/7010-series jobs under operating system control offers emulation users the benefits of multiprogramming and OS facilities. The advantages of Model 145 integrated emulation are:

- Significantly better resource utilization, since System/370 native mode and 1400/7010-series emulator jobs can be multiprogrammed. Stand-alone emulators normally use only a portion of the hardware resources available in the system.

- Standardized and simplified job accounting and job scheduling. The latter reduces the number of IPL's required because switching from operating system to stand-alone emulation mode of system operation is not required.

- The ability to extend or add applications such as graphics, teleprocessing, time sharing, etc., because a dedicated emulation environment is no longer required and more system resources are available in a given time period.

- The ability to process certain 1400/7010-series tape and disk data sets, using both System/370 and emulated 1400/7010-series programs. Existing 1400/7010 tape files can be converted to a standard OS data format using the IBM-supplied Tape Preprocessor formatting program. Existing 1400/7010 disk files must be converted.

- More efficient use of direct access space, since both 1400/7010 and System/370 data sets can be placed on the same disk volume.

- Device independence for emulator-supported I/O devices used by emulated 1400/7010-series programs that handle data sets in OS variable-length spanned (VS) or variable-length spanned blocked (VBS) data format. OS access methods are used to handle I/O operations so that new functions and I/O device support added to the access method routines used by the emulator programs are automatically available to emulated 1400/7010-series jobs. Tape and unit record 1400/7010-series files can be emulated on System/370 direct access devices.

The two emulator programs provided for the Model 145 use simulation routines, compatibility feature microprogram instructions, the Model 145 instruction set, and OS supervisor and data management routines to emulate 1400/7010-series program operations. QSAM is used for unit record emulation, BSAM is used for tape emulation, and BDAM is used for disk emulation. Figure 40.05.1 shows the general layout of an emulator program partition or region and indicates the range of processor storage requirements for both emulator programs. Note that 1400/7010 storage is simulated in contiguous Model 145 storage (each 1400/7010 position is simulated by a byte).

1401/1440/1460	Partition or Region	1410/7010
2K to 10K	OS Data Management Routines	2K to 10K
	Buffers and Control Blocks	
	Available Storage	
2K to 16K 1401 Systems 2K to 16K 1440 Systems 8K to 16K 1460 Systems	Simulated 1400/7010 Storage (size of system being emulated)	10K to 80K 1410 Systems 40K to 100K 7010 Systems
Approximately 21K to 36K	Emulator Routines	Approximately 23K to 46K

Figure 40.05.1. Partition or region layout for an OS 1400/7010-series emulator program job step, with general storage requirements indicated

The specific emulator program to be used by an installation must be constructed via an emulator generation procedure, which produces the job control statements required to assemble and link-edit the desired emulator modules and place the emulator program in SYS1.LINKLIB. Emulator program routines and data formatting programs (Tape Preprocessor, Tape Postprocessor, and Format Disk) are distributed on a restore tape independently from regular OS releases. All Model 145 1400/7010-series emulator users receive the same two tape formatting programs. One disk formatting program is supplied. It handles preformatting for 1401/1440/1460 files and 1410/7010 files. The following must be done to include one of the emulators in an OS operating system for the Model 145:

- Certain facts about the emulator program to be used with the operating system generated must be specified in the input required to generate the OS control program.

- The emulator restore tape must be obtained from PID, and an emulator program with the desired facilities must be generated and placed in SYS1.LINKLIB. More than one version of a 1400/7010-series emulator program for the same and different systems can exist in an operating system.

The emulator program generated will emulate, without change, 1400/7010 programs that are written in accordance with IBM 1400/7010 Principles of Operation manuals and that are operating on 1400/7010 systems, subject to the following conditions:

1. Time-dependent programs may not execute properly. Provision has been made to allow some time-dependent programs to be emulated correctly. (See the appropriate emulator planning manual for details.)

2. Programs that depend on error conditions or on the absence of a particular feature may not be emulated correctly.

3. Programs with undetected programming errors will give unpredictable results.

4. Only the 64-character BCD set is accepted by the emulators.

5. Programs that use unsupported features or I/O devices (as described for each emulator in this subsection and in the emulator program reference manuals) must be modified to conform to the support provided by the specific emulator program, unless a user routine is written to handle the feature.

The Model 145 1400/7010 integrated emulator programs support the same facilities as the stand-alone 1401/1440/1460 and 1410/7010 emulators for System/360 models except for a few special features not supported by the Model 145 emulators. Thus, any 1400/7010 program that is being executed by a stand-alone System/360 emulator and that does not use one of these special features can be emulated on the Model 145 without change.

Tape and Disk Formatting Programs and Data Formats

The Tape Preprocessor and Tape Postprocessor formatting programs supplied to Model 145 1400/7010-series emulator users operate as processing programs and can be executed with any OS control program generated with the emulator macro specified. The Tape Preprocessor operates in a program area of 4K bytes plus I/O buffer requirements and accepts as input seven- and nine-track tape in 1400/7010-series format with or without 1400/7010 labels. It accepts mixed density 1400/7010 files, that is, files with header labels written in a density different from that of the data. The emulator programs consider a change in density to be an error. Therefore, mixed density 1400/7010-format files must be preprocessed.

The preprocessor produces as output spanned variable-length (OS VS or VBS) format data that can be written on seven- or nine-track tape or on direct access storage. Input records longer than 32,755 bytes are reblocked, since OS BSAM cannot handle a physical data block longer than 32K. VS or VBS format tape can be unlabeled or have OS standard labels, in addition to any 1400/7010 labels. The preprocessor converts 1400/7010 labels and tapemarks into data records that are recognized by the emulator program. Thus, if VS or VBS format tapes with 1400/7010 label data records are to be processed by System/370 programs, the tape must be rewritten to remove the label data records or the System/370 program must contain a routine to recognize these records.

The Tape Postprocessor operates in a program area of 5K bytes plus I/O buffer requirements and performs the reverse of the Tape Preprocessor. The postprocessor program is useful when a copy of a data set in OS VS or VBS and another in 1400/7010 format are required or if mixed density 1400/7010-format files are required. (The 1400/7010-series emulator programs accept as input and produce as output both the formats handled by these two tape formatting programs.)

The tape formatting and emulator programs handle 200-, 556-, 800-, and 1600-BPI density, mixed-density, and even-, odd-, and mixed-parity seven-track tapes. VS, VBS, or 1400/7010-format data written on nine-track tape is coded in EBCDIC. If VS or VBS format tapes are processed on a seven-track tape unit, the tape control unit must have seven-track and data convert features installed. The alternate mode used by stand-alone System/360 1400/7010 emulators is accepted by the Model 145 emulator programs as well.

While existing tape files with blocks longer than 32K bytes must be preprocessed, conversion to VS or VBS format offers the following advantages:

- The ability to emulate tape data sets on direct access devices for more flexibility in I/O device assignment

- The ability to increase emulator job performance by reblocking 1400/7010-series-format tape files with short blocks

- The ability to reduce processor storage buffer requirements by reblocking files with very large blocks

- The ability to process VS or VBS format tape and move mode disk data sets with both OS and emulated 1400/7010-series programs if the OS programs can handle 1400/7010 label and tapemark records. (Load mode disk data sets are not accepted by OS programs.)

The disk formatting program supplied operates as a processing program, and an area of 2K bytes in addition to buffer requirements is needed for its execution. This program must be used to format System/370 disk volumes that are to contain 1400/7010 disk data. In order to convert 1400/7010 disk files that are being processed on a 1400/7010 system or by a stand-alone System/360 1400/7010 emulator to a format acceptable to the Model 145 emulator program, the following must be done:

1. The contents of the disk must be dumped to tape, using a 1400/7010 disk-to-tape utility program. This must be done on a 1400/7010 system if the data was created and is being processed on a 1400/7010 system. A System/360 with a stand-alone 1400/7010 emulator must be used to convert 1400/7010-format data files that are being emulated on System/360 direct access devices.

2. One or more initialized System/370 disk volumes must be formatted by executing the IBM-supplied disk formatting program on a Model 145 under OS MFT or MVT control.

3. A 1400/7010 tape-to-disk utility program must be executed on a Model 145 under emulator program control to restore the 1400/7010 data on tape to the formatted System/370 disk volume(s).

The data records on one 1400/7010 disk track are formatted into one fixed-length record, which is generally placed on one System/370 disk track. If one System/370 track is not large enough to contain the fixed-length record created from one 1400/7010 track, the Track Overflow feature is required on the System/370 device or another System/370 disk device type must be used. Depending on the disk devices involved, one System/370 disk track may be large enough to contain the data from more than one 1400/7010 disk track, and use of the Track Overflow feature is an option. If used, it may result in more efficient use of System/370 direct access space. However, I/O processing time is increased by use of the Track Overflow feature.

Job Submission and Operator Communication

OS job control and Model 145 1400/7010 Emulator program control statements must be present for emulated 1400/7010-series object programs. Several 1400/7010 programs can be emulated in one OS job step. Subject to the restrictions listed previously, existing 1400/7010 programs need not be modified.

The required emulator control statements for emulated 1400/7010 programs are provided in a card, tape, or disk sequential data set or in the input stream. The 1400/7010 object programs to be emulated can be placed in the input stream, on tape, or in a partitioned data set on disk. An emulator control statement describes their location.

Card input to 1400/7010 programs can be placed in the input stream to be read by the reader/interpreter and placed in a SYSIN data set. Alternatively, card input can be emulated via a tape or disk sequential data set. Data to be printed or punched can be placed in a sequential tape or disk data set or in a SYSOUT data set on disk for transcription by an output writer. Use of an OS reader interpreter and output writer to handle unit record operations should reduce the execution time required to emulate 1400/7010 programs.

The operator communicates with an emulator partition or region via emulator commands that can be entered using the operating system console. The commands provided allow simulation of 1400/7010 console operations and can also be used in debugging operations. The operator can display I/O assignments, sense switch settings, etc., in effect for an executing 1400/7010 program. In addition, the operator can alter and dump processor storage selectively within the emulator program partition or region.

If multiple console support (MCS) is included in the OS control program generated, emulator program messages can be routed to a specific console device so that emulation messages are isolated. Automatic reply routines that process 1400/7010 program messages can be user-written and added to the generated emulator program. This facility can be used to avoid having 1400/7010 program messages printed on a console.

Installing an Emulator

The following outlines the general procedure required to make the transition from 1400/7010 system operation or stand-alone System/360 1400/7010 emulation to Model 145 emulation under OS.

1. Generate an OS operating system for the Model 145, and specify required parameters that describe the 1400/7010 emulator program to be used with this operating system. (Generating an operating system for the Model 145 is discussed in Section 60.)

2. Generate the desired 1400/7010 emulator program.

3. Add required OS job control and emulator control statements to the existing 1400/7010 programs that are to be emulated on the Model 145. Subject to the conditions stated previously in this section, modification of 1400/7010 programs may or may not be required. Optionally, 1400/7010 programs can be placed in a library (PDS) by using the OS IEBUPDTE utility program.

4. Tapes in 1400/7010 mode with blocks shorter than 18 bytes or longer than 32K must be preprocessed.

5. Disk files must be transferred to Model 145 direct access devices by using the steps already outlined.

6. Optionally, routines to support features and I/O devices not handled by the emulator program can be written and placed in a library. The generated 1400/7010 emulator program will cause them to be loaded.

When installation of a 1400/7010 emulator is being planned, consideration should be given to the factors that affect the performance of emulated 1400/7010 jobs. Throughput of 1400/7010 jobs is affected by the mix of CPU and I/O operations executed by the compatibility feature and the amount of interference from higher priority partitions or regions. A large factor in performance is the way I/O operations are handled. The following steps can be taken to achieve improved emulator job throughput if enough processor storage is available:

1. Allocate one buffer to each access mechanism simulated instead of sharing buffers among multiple access mechanisms. (Do not use the shared buffer option for disk data sets unless processor storage is limited.)

2. Allocate two buffers to each tape data set instead of one.

3. Convert 1400-format tape files to VS or VBS format for emulator processing.

4. Using the Tape Preprocessor, reblock tape files containing short blocks.

5. Do not use the Track Overflow feature for emulated disk data sets unless direct access space is at a premium.

OS 1410/7010 EMULATOR PROGRAM SUPPORT

The OS 1410/7010 Emulator program can emulate a small 1410 system on a 160K Model 145 with the 1401/40/60, 1410/7010 Compatibility feature and enough I/O devices for the operating system and emulated 1410/7010 devices.

The partition or region size required depends on the features, I/O devices, and buffering used and on the size of the system being emulated. The emulator program size varies from a minimum of approximately 23,500 bytes, for a basic system with unit record I/O device support only, to a maximum of approximately 43,000 bytes (1410) or 46,600 (7010) for all special features and unit record, tape, and disk support. A large 7010 system--100K, 14 tape units, each with a 1K buffer, 4 disk units using two 2K shared buffers, and unit record devices with 400 bytes of buffers--requires a 166K partition for emulation. This figure includes 7500 bytes of access method routines. (For details about processor storage requirements, see 1410/7010 OS Emulator on Models 145/155, GC33-2009.)

Table 40.05.1 lists the 1410/7010 system features supported and not supported by the Model 145 1410/7010 Emulator program. Table 40.05.2 lists 1410/7010 I/O devices and features emulated and their Model 145 I/O device counterparts, while Table 40.05.3 indicates unsupported 1410/7010 devices.

The number of Model 145 direct access devices required to emulate a 1410/7010 disk device is indicated in Table 40.05.4. Requirements with and without use of the Track Overflow feature are indicated. Note that 2311 drives without track overflow cannot be used to emulate 1302 and 2302 Disk Storage.

The Model 145 integrated OS 1410/7010 Emulator program supports the same facilities and I/O devices as the stand-alone 1410/7010 emulator for System/360 models except for Stacker Selection on the 1402 Card Read Punch.

Table 40.05.1. 1410/7010 system features supported and unsupported by the Model 145 OS 1410/7010 Emulator program

1410 Features	7010 Features
Supported	Supported
All basic CPU functions Core storage up to a maximum of 80,000 positions Inverted Print Edit Priority Processing Processing Overlap One or two channels	Standard 7010 instruction set with store and restore status Main storage up to a maximum of 100,000 positions Floating Point Arithmetic Processing Overlap Priority Feature Inverted Print Edit One to four channels
Unsupported	Unsupported
The 1400 Diagnostic instruction Branch on Tape Indicate	1401/1410 Compatibility Mode Diagnostic instruction Branch on C Bit Program Relocation Storage Protection Interval Timer

Table 40.05.2. 1410/7010 I/O devices and features emulated by the OS 1410/7010 Emulator program and their Model 145 equivalents

1410/7010 I/O Device	Model 145 I/O Device
• 1402 Card Read Punch and 51-Column card feature 1442 Card Reader Features not emulated are: Select Stacker Column Binary Programmed reading from more than one reader or punching on more than one punch within a program is not supported.	• Any card reader, card read punch, magnetic tape unit, or direct access device supported by OS QSAM
• 1403 Printer Programmed printing on more than one printer within a program is not supported.	• Any printer, magnetic tape unit, or direct access device supported by OS QSAM. A printer must have the UCS feature to emulate the preferred character set and numerical print features correctly.
• 729 Model II, IV, V, and VI Magnetic Tape Units 7330 Magnetic Tape Unit	• Any tape unit or direct access device supported by OS BSAM. VS or VBS format is used for 1410/7010 tape files emulated on a direct access device. Seven-track tapes must be emulated on tape units with a seven-track head. The tape control unit must have seven-track and data convert capability if EBCDIC is used.
• 1415 Console	• Any Model 145 operator console supported by OS
• 1301 Disk Storage Model 1 or 2 1302 Disk Storage Model 1 or 2 2302 Disk Storage Model 1 or 2 All features emulated except Write Disk Check (treated as a no-op) and operations involving the CE tracks. Write verification can be requested in the OS job control statement for the emulated disk device.	• Any direct access storage device supported by OS BDAM. A maximum of 20 simulated arms per channel are supported. If two or more System/370 direct access devices are required to emulate one 1410/7010 disk device, all Model 145 disk devices used to emulate that device must be of the same type. The 2311 cannot be used to emulate 1302 or 2302 Disk Storage unless the Track Overflow feature is present.

Table 40.05.3. 1410/7010 I/O devices not supported by the Model 145 OS 1410/7010 Emulator program

1311 Disk Storage Drive	Magnetic character readers
1405 Disk Storage	7340 Hypertape Drive
1011 Paper Tape Reader	Teleprocessing devices
1012 Tape Punch	Optical character readers

Table 40.05.4. Model 145 direct access device requirements for emulation of 1410/7010 disk devices using OS with and without the Track Overflow (T.O.) feature.

1410/7010 Disk Device	Model 145 Disk Devices Number of Cylinders Required per 1410/7010 Device					
	With T.O.			Without T.O.		
	2311	2314	3330	2311	2314	3330
1301 Model 1	819	208	121	1000	250	132
1301 Model 2	1638	416	242	2000	500	264
1302/2302 Model 1	3402	840	486	-	1000	528
1302/2302 Model 2	6804	1680	972	-	2000	1056

OS 1401/1440/1460 EMULATOR PROGRAM SUPPORT

The OS 1401/1440/1460 Emulator program can operate on a Model 145 with 112K or more of storage, the 1401/1440/1460 Compatibility or the 1401/40/60, 1410/7010 Compatibility feature, and enough I/O devices for the operating system and emulated 1400 series devices.

The partition or region size required depends on the features, I/O devices, and buffering used and on the size of the system being emulated. The emulator program size varies from a minimum of approximately 21K, for a basic system with unit record I/O device support only, to a maximum of approximately 36K for all special features and unit record, tape, and 1311/1301 disk support. A 16K 1401 system with unit record devices (400 bytes of buffers) and six tape units (1K buffer per tape unit) can be emulated in a 54K partition. This figure includes approximately 3.9K for access method routines. (For details about processor storage requirements see 1401/1440/1460 OS Emulator on Models 135/145/155, GC33-2008.) Note that a 1401/1440/1460 Emulator program can be generated to handle 1405 Disk Storage or 1311/1301 Disk Storage, but not both.

Table 40.05.5 lists the 1401/1440/1460 system features supported and not supported by the Model 145 OS 1401/1440/1460 Emulator program. Table 40.05.6 lists 1401/1440/1460 I/O devices and features emulated and their Model 145 I/O device counterparts, while Table 40.05.7 indicates unsupported 1400-series devices. The number of Model 145 direct access devices required to emulate a 1400-series disk device is indicated in Table 40.05.8. Requirements with and without use of the Track Overflow feature are indicated.

The Model 145 integrated OS 1401/1440/1460 Emulator program supports the same facilities and I/O devices as the stand-alone 1401/1440/1460 emulator for System/360 models except for Punch Feed Read and Stacker Selection on the 1402 Card Read Punch and Column Binary and Binary Transfer in the CPU.

Table 40.05.5. 1401/1440/1460 system features supported by the Model 145 OS 1401/1440/1460 Emulator program

All basic CPU operations
 1401 Models A-F, H
 1440 Models A2-A6
 1460 all models
Core storage up to 16,000 positions
Expanded Print Edit
Inverted Print Edit
High-Low-Equal Compare
Multiply-Divide

Sense Switches
Indexing and Store Address Register
Advanced Programming
Bit Test
Print Storage
Additional Print Control
Space Suppression
Processing Overlap

Table 40.05.6. 1401/1440/1460 I/O devices and features emulated by the OS 1401/1440/1460 Emulator program and their Model 145 counterparts

1401/1440/1460	Model 145 I/O Device
• 1402 Card Read Punch (1401, 1460) and 51-Column Card feature 1442 Card Read Punch (1440) 1442 Card Reader (1440) The following are not emulated: Column Binary (1402) Punch-Feed Read (1402) Read-Punch Release (1402) Punch-Column-Skip (1442) Read and punch same card (1442) Punch and Stop (1442) Stacker selection (treated as a no-op except on the 2540) Programmed reading from more than one reader or punching on more than one punch within a program is not supported.	• Any card reader, card read punch, magnetic tape unit, or direct access device supported by OS QSAM. Stacker selection is supported only on the 2540.
• 1443, 1403 Printer The Selective Tape Listing feature and programmed printing on more than one printer are not supported.	• Any printer, magnetic tape unit, or direct access device supported by OS QSAM. A printer must have the UCS feature to emulate the preferred character set and numerical print features correctly.
• 729 Model II, IV, V, and VI Magnetic Tape Units 7330, 7335 Magnetic Tape Units (Compressed tapes are not supported.)	• Any tape unit or direct access device supported by OS BSAM. VS or VBS format is used for 1400 tape files emulated on a direct access device. Seven-track tapes must be emulated on tape units with a seven-track head attached to a tape control unit with seven-track and data convert capability
• 1407 Console Inquiry Station 1447 Console The Branch on Buffer Busy feature (1447) is not supported.	• Any Model 145 operator console supported by OS

Table 40.05.6. (continued)

1401/1440/1460	Model 145 I/O Device
• 1301 Disk Storage (only one access arm) 1311 Disk Storage Drive Models 11 and 12 1405 Disk Storage Models 1 and 2 Up to five 1311 drives and one 1301 module or one 1405 Model 1 or 2 can be emulated. All features are supported except Write Disk Check (treated as a no-op). Write verification can be requested in the OS job control statement for the emulated disk device.	• Any direct access device supported by OS BDAM. If two or more System/370 direct access devices are required to emulate one 1400 disk device, all Model 145 disk devices used to emulate that device must be of the same type.

Table 40.05.7. 1401/1440/1460 I/O devices and features not supported by the Model 145 OS 1401/1440/1460 Emulator program

1400 I/O Device	1400 Feature
1404 Printer 1444 Card Punch 1445 Printer 1011 Paper Tape Reader 1012 Tape Punch Optical readers Magnetic character readers 7340 Hypertape Drive Teleprocessing devices	1401 Model G Binary Transfer Column Binary Translate (1460)

Table 40.05.8. Model 145 direct access device requirements for emulation of 1401/1440/1460 disk devices using OS with and without the Track Overflow (T.O.) feature.

1401/1440/1460 Disk Device	Model 145 Disk Drives Number of Cylinders Required per 1400 Device					
	With T.O.			Without T.O.		
	2311	2314	3330	2311	2314	3330
1405 Model 1	316	82	47	334	84	48
1405 Model 2	632	163	93	667	167	96
1311 (Sector mode)	64	17	9	100	17	10
1311 (Track mode or both track and sector mode)	87	22	12	100	25	13
1301 (Sector mode)	634	161	92	1000	167	105
1301 (Track mode or both track and sector mode)	744	188	108	1000	250	132

40:10 DOS 1401/1440/1460 AND 1410/7010 EMULATOR PROGRAMS

FEATURES COMMON TO BOTH EMULATORS

The Model 145 continues the advantages of integrated emulation for DOS CS/30 and CS/40 users. In addition, these advantages are now extended to users of 1410/7010 stand-alone emulation.

A 1401/1440/1460 and a 1410/7010 Emulator program are provided. Both emulator programs are common in design and run as problem programs under DOS Versions 3 and 4, and DOS/VS on a Model 145 equipped with either the 1401/1440/1460 Compatibility or the 1401/40/60, 1410/7010 Compatibility feature. The latter feature permits operation of both the 1401/1440/1460 emulator and the 1410/7010 emulator. These two no-charge compatibility features are also used by the Model 145 OS 1401/1440/1460 and 1410/7010 emulators.

As discussed at the beginning of Section 40:05 (OS 1400/7010 emulator discussion), a DOS 1400/7010 emulator program generated for the Model 145 will run on a Model 135 (1401/1440/1460 only) but not on a Model 155, while a DOS 1400/7010 emulator program generated for a Model 155 will also run on Models 135 (1401/1440/1460 only) and 145.

The emulators can be used in a batch-only system environment or can operate in the background and batched foreground partitions of a multiprogramming system. Therefore, more than one 1401/1440/1460 or 1410/7010 Emulator program can execute concurrently with each other and with System/370 programs. Additionally, emulated jobs and DOS jobs can be intermixed in a single job stream.

The Model 145 DOS integrated emulators consist of a compatibility feature, simulation routines, and DOS data management routines. They offer Model 145 users the following advantages:

- System resources are more fully utilized.

- Emulators can run concurrently in all three partitions of a multiprogramming system. They are relocatable and can be link-edited to run in any partition.

- 1401/1440/1460 and 1410/7010 Emulator programs and DOS programs can be executed concurrently and intermixed in a single job stream.

- DOS supervisor and data management services are available to the user. This provides job control facilities, standard disk and tape label processing, and common data formats for emulator files and DOS files.

- 1400/7010 unit record input/output operations can be made device independent and can be emulated on Model 145 unit record devices, magnetic tape units, or direct access storage devices.

Emulator Program Generation and Execution

The Model 145 DOS emulators are distributed with DOS releases. An emulator is assembled by the use of macro instructions. The macro instructions describe the 1400/7010 CPU, input/output devices, special features, data files, emulator buffers, and the desired user options. When assembled, the macros provide an object module and linkage to preassembled modules stored in the system relocatable library. The preassembled modules are combined with the emulator object module by the linkage editor for cataloging in the core image library. Any number of emulators can be assembled and cataloged in a core image library to run in any partition.

The emulator program generated will emulate, without change, 1400/7010 programs written in accordance with IBM 1400/7010 Principles of Operation manuals, subject to the following conditions:

- 1400/7010 programs that purposely depend on the absence of a 1400/7010 feature or on error conditions may not execute properly.

- Programs with undetected programming errors and those that depend on timing of 1400/7010 I/O operations yield unpredictable results.

An emulator program is handled by DOS in the same manner as any problem program. When using the 1401/1440/1460 emulator, 1400 programs may be cataloged to, and fetched from, a core image library for execution or loaded from a card, tape, or direct access storage device; 1410/7010 programs can be loaded only from a card, tape, or direct access storage device. Standard DOS job control statements are used to prepare the system for an emulator job. The EXEC job control statement causes the specified emulator program to be loaded, and control is passed to the emulator program. Emulator control statements are read by the emulator from the logical unit selected at generation time.

Emulation with the 1401/1440/1460 and 1410/7010 emulators consists of three main steps:

1. Initialization. Emulator control statements supplied by the user are read and interpreted. This information overrides, for the execution of the emulator, information specified at emulator generation.

2. Loading (or fetching - 1401/1440/1460). The 1400/7010 program is loaded from a card reader, magnetic tape, or direct access storage device. For the 1401/1440/1460 emulator, a 1400 program can be fetched from a core image library if it has been cataloged.

3. Execution (or precataloging - 1401/1440/1460). When loaded, the 1400/7010 program is executed. The 1400/7010 instructions are fetched, interpreted, and executed by the emulator until an end-of-job condition is recognized. For the 1401/1440/1460 emulator, the 1400 program can be either executed or converted into a DOS object module (precataloged). This module can be subsequently link-edited and cataloged in a core image library.

Input/output errors are processed by DOS device error recovery procedures. Input/output errors that cannot be corrected, such as permanent input/output errors and wrong-length records, are passed to the 1400/7010 program.

Console simulation and operator communication with the emulators are provided by the exchange of emulator commands and messages between the operator and the emulators. The emulators provide messages to inform the operator of errors or other conditions that require his attention or a response. Emulator commands can be entered from the console printer-keyboard and are handled in the same way as operator communications are currently handled by DOS.

Tape and Disk Emulation

The user has the option of processing tape files in 1400 format or in spanned variable-length record format. Two tape formatting programs--the Tape Preprocessor and Tape Postprocessor--are available to convert tape files from 1400 format to spanned record format, and vice versa; 1410/7010 tapes containing records larger than 32K must be converted to spanned record format prior to emulation. Mixed-density tapes are not supported by the emulators or the tape formatting programs.

The emulators accept as input and produce as output two tape file formats:

1. 1400 format, which is produced by a 1400/7010 system, a stand-alone emulator, CS/30, CS/40, the Tape Postprocessor formatting program, or the System/370 1400 emulators

2. Spanned variable-length record format, which is produced by the Tape Preprocessor formatting program or a System/370 1400 emulator

Processing tape files in spanned variable-length record format provides several advantages:

- Blocking short records reduces the time for emulating I/O operations.

- The Tape Preprocessor or the Tape Postprocessor program can be run concurrently with the emulators in a multiprogramming system.

- Files in spanned variable-length record format can be used by other System/370 programs if the programs provide for handling the 1400/7010 label records and 1400/7010 tapemark records.

- The Tape Postprocessor program can be used to convert a file in spanned variable-length record format back to 1400 format for use on a 1400/7010 system.

Tape files in spanned record format have standard DOS labels; 1400/7010 labels are treated as data records, since they are processed by the 1400/7010 program. The 1400/7010 tapemarks appear as special data records and are recognized by the emulators.

The character codes supported by the emulators for magnetic tape data are:

- BCD representation in even and odd parity for seven-track tape (data translator on) in 1400 format

- BCDIC-8 representation for nine-track tapes in either 1400 or spanned record format, and for seven-track tapes (data converter on) in spanned record format. This character code, which is the eight-bit representation of BCD, is used to simulate parity. In normal mode, bit 1 is set to one for even parity, to zero for odd parity. In alternate mode, bit 1 is always set to one and no distinction is made between even and odd parity.

Two tape formatting programs, a preprocessor and a postprocessor, are available for converting tape files. They are distributed to run as problem programs under DOS control. They can be executed only in the background partition. The Tape Preprocessor program converts seven-track or nine-track tapes in 1400 format to seven-track (data converter on) or nine-track tapes in spanned variable-length record format with standard DOS labels. The Tape Postprocessor converts seven-track or nine-track tapes in spanned record format to seven-track or nine-track tapes in 1400 format.

The DOS Clear Disk utility program is required to preformat Model 145 disk packs that are used for emulation of 1400 disk files. Each Model 145 disk record represents one 1400 disk track. Each Model 145 disk record is a fixed-length record, its length being a function of the emulated 1400/7010 device and mode rather than the amount of 1400/7010 data on each track. A 1400/7010 disk file can occupy one or more extents on Model 145 disk packs but only one extent per pack. Extents must be allocated complete cylinders. When a file requires more than one Model 145 disk pack, the packs must be the same type. Two different

1400/7010 files may be placed on the same disk pack, but this arrangement may increase seek time if both files are processed at the same time.

Character codes supported by the emulators for disk files are:

- EBCDIC representation for disk operations in move mode

- BCDIC-8 representation for disk operations in load mode. (Data written in load mode must be converted to EBCDIC if it is to be used by programs other than the emulators.)

Disk files in 1400 format, which are created on a 1400/7010 system or under stand-alone emulation, must be converted to a standard fixed-length record format on 2311, 2314-type, or 3330-series disk packs before emulation. Disk files created under CS/30 and CS/40 can be processed by the 1401/1440/1460 emulator if the CS option is specified at emulator generation.

To convert disk files in 1400 format, or CS/30 and CS/40 disks if desired, to a standard format on a Model 145 disk, the user must dump and restore his data as follows:

1. Dump the disk device, using a 1400/7010 disk-to-tape or disk-to-card utility program. When converting files on 1301, 1311, 1405, or 2302 disk devices that were created on a 1400/7010 system, the utility is executed on the system used to create the file. When converting files on 2302, 2311, or 2314 disks that were created under stand-alone emulation, CS/30, or CS/40, the utility is executed on a System/360 under control of the emulator used to create the disk file.

2. Use the DOS Clear Disk utility program to format previously initialized 2311, 2314-type, or 3330-series disk pack(s) for the data.

3. Restore the data to a formatted 2311, 2314-type, or 3330-series disk pack, using a 1400/7010 tape-to-disk or card-to-disk utility program under control of a Model 145 emulator. The 1401/1440/1460 emulator is used to restore 1401/1440/1460 data; the 1410/7010 emulator to restore 1410/7010 data.

DOS 1401/1440/1460 EMULATOR SUPPORT

For the DOS CS/30 and CS/40 user, the Model 145 DOS 1401/1440/1460 emulator continues the advantages of integrated emulation and provides additional advantages, such as:

- Emulators operating concurrently in all three partitions (now a restriction for the CS/30 user)

- Simulated 1400 storage that can begin at any address (CS/40 users are restricted to beginning addresses that are multiples of 16K for simulated 1400 storage)

- Support of CS/30 and CS/40 disk and tape files and emulator control statements

- System/370 data formats for emulator tape and disk files

- Added emulator control available to user at execution time

- DOS data management facilities and standard disk and tape label processing

The size of the partition required for emulation depends on the 1400 system being emulated, including standard and special features, input/output devices, buffers, etc. The processor storage required for the 1401/1440/1460 emulator is equal to the combined sizes of:

- Simulated 1401/1440/1460 storage. Each position of 1400 storage is simulated in one byte of Model 145 storage (for example, 8000 positions = 8000 bytes).
- Emulator routines required to emulate the 1401/1440/1460 system instructions, features, and I/O operations
- Tape, disk, and unit record buffers. The number and size of tape and disk buffers are specified by the user.

Approximate minimum 1401/1440/1460 emulator processor storage requirements for emulation of a 1400 system with unit record operations only, unit record/tape operations, or unit record/tape/disk operations are shown below. For details about processor storage requirements, see 1401/1440/1460 DOS Emulator on Models 135/145/155 (GC33-2004).

Emulated Operations	DOS Partition Size (bytes)
1401/1440/1460 unit record	18K + 1401/1440/1460 storage size + buffers
1401/1440/1460 unit record and 6 tapes	23K + 1401/1440/1460 storage size + buffers
1401/1440/1460 unit record, 6 tapes, 4 disks	27K + 1401/1440/1460 storage size + buffers

The 1400 CPU features and 1400 I/O devices and special features supported and the Model 145 devices used for 1401/1440/1460 emulation are given in Tables 40.10.1 and 40.10.2. Table 40.10.3 lists the 1400 I/O devices that are not supported.

Emulator performance will vary depending on user options, such as number and size of buffers, the instruction mix of the 1401/1440/1460 programs, the format of tape files, and the priority of the partition in which the emulator is running.

Emulator performance is improved by:

1. Using double buffers and spanned record format for tape files in lieu of single or shared buffers and 1400 record format. (A shared buffer can be used by more than one I/O device.)
2. Using single buffers rather than shared buffers for disk files
3. Specifying device independence for emulating unit record operations on a magnetic tape or direct access storage device
4. Generating the emulator without support for the 51-Column Interchangeable Read Feed and Column Binary features and the Select Stacker instruction

Table 40.10.1. 1401/1440/1460 I/O devices and features supported by the DOS 1401/1440/1460 Emulator program and corresponding Model 145 devices

1401/1440/1460 Device and Features	Corresponding Model 145 Device
1402, 1442, 1444 Card Read Punch with stacker selection Features supported: • Column Binary or Card Image • 51-Column Interchangeable Read Feed • Punch Feed Read • Punch Column Skip • Binary Transfer • Processing Overlap • Read Punch Release (as a no-op) Not supported: • Multiple card reader/punch operations in one program	1442, 2520, 2540 Card Read Punch, 2501 Card Reader, 3505 Card Reader, 3525 Card Punch Note: Card reader and card punch operations may be emulated using a magnetic tape or direct access storage device.
1403, 1404, 1443 Printer Features supported: • Numerical Print • Processing Overlap • Space Suppression Not supported: • Selective Tape Listing • Multiple printer operations • Cut-card operations and Read Compare	1443, 1403, 3211 Printer Note: Printer operations may be emulated using a magnetic tape or direct access storage device.
1407, 1447 consoles	3210 or 3215 console
729, 7330, 7335 Magnetic Tape Unit Features supported: • Binary tape instructions • Processing Overlap Features not supported: • Compressed tape	2400- and 3400-series magnetic tape units • Seven-Track feature is required if processing seven-track tapes
1301, 1311, 1405 Disk Storage Features supported: • Direct Seek • Scan Disk • Track Record • Additional access arm (1405) Note: A 1405 cannot be emulated in combination with a 1301 or 1311	2311, 2314-type, 3330-series direct access devices

Table 40.10.2. 1401/1440/1460 CPU features supported by the DOS 1401/1440/1460 Emulator program

Storage from 1400 to 16,000 positions.
Expanded Print Edit
Inverted Print Edit
High-Low-Equal Compare
Move Binary Code and Decode
Multiply-Divide
Sense Switched
Advanced Programming
Indexing and Store Address Register
Bit Test

Note: Translate feature is not supported.

Table 40.10.3. 1401/1440/1460 devices not supported by the DOS 1401/1440/1460 Emulator program

1445 Printer
Paper tape readers
Paper tape punches
Magnetic character readers
Optical character readers
1404 Printer in cut card mode
7340 Hypertape Drive
Teleprocessing devices
Audio response units

DOS 1410/7010 EMULATOR SUPPORT

The Model 145 DOS 1410/7010 emulator offers the 1410/7010 stand-alone emulator user the advantages of integrated emulation already discussed.

Processor Storage Requirements

The size of the partition required for emulation depends on the 1410/7010 system being emulated, including standard and special features, input/output devices, buffers, etc. The processor storage required for the 1410/7010 emulator is equal to the combined sizes of:

- Simulated 1410/7010 storage. Each position of 1410/7010 storage is simulated in one byte of Model 145 storage (for example, 20,000 positions = 20,000 bytes).

- Emulator routines required to emulate the 1410/7010 system instructions, features, and I/O operations

- Tape, disk, and unit record buffers. The size and number of tape and disk buffers are specified by the user.

Approximate minimum 1410/7010 emulator processor storage requirements for emulation of a 1410/7010 system with unit record/tape operations or unit record/tape/disk operations are shown below. For details about processor storage requirements, see 1410/7010 DOS Emulator on Models 145/155 (GC33-2005).

Emulated Operations	DOS Partition Size (bytes)
1410/7010 unit record and 6 tapes	25K + 1410/7010 storage size + buffers
1410/7010 unit record, 6 tapes, 4 disks	34K + 1410/7010 storage size + buffers

The 1410/7010 CPU features and 1410/7010 I/O devices and special features supported and the Model 145 devices used for 1410/7010 emulation are given in Tables 40.10.4 and 40.10.5. Table 40.10.6 lists the 1410/7010 I/O devices that are not supported.

Emulator performance will vary depending on user options, such as number and size of buffers, the instruction mix of the 1410/7010 program, the format of tape files, and the priority of the partition in which the emulator is running.

Emulator performance is improved by:

1. Using double buffers and spanned record format for tape files in lieu of single or shared buffers and 1400 record format. (A shared buffer can be used by more than one I/O device.)

2. Using single buffers rather than shared buffers for disk files

3. Emulating unit record operations on a magnetic tape or direct access storage device

Table 40.10.4. 1410/7010 I/O devices and features supported by the DOS 1410/7010 Emulator program and corresponding Model 145 devices

1410/7010 Device and Features	Corresponding Model 145 Device
1402, 1442 Card Read Punch Features not supported: Stacker selection Column Binary 51-Column Interchangeable Read Feed Multiple card read/punch units in one program	1442, 2540, 2520 Card Read Punch 2501 Card Reader, 3505 Card Reader, 3525 Card Punch Note: Card reader and card punch operations may be emulated using a magnetic tape or direct access storage device.
1403 Printer All standard operations Multiple printers in one program are not supported.	1443, 1403, 3211 Printer Note: Printer operations may be emulated using a magnetic tape or direct access storage device.
1415 Console All standard operations	3210 or 3215 console
729, 7330 Magnetic Tape Units All standard operations	2400- and 3400-series magnetic tape units Note: Seven-Track feature is required for processing seven-track tapes.
1301, 1302, 2302 Disk Storage All standard operations Note: Any combination of 1301 and 1302/2302 disk storage drives can be emulated.	2311, 2314-type, 3330-series direct access devices Note: An emulated 1302/2302 record will not fit on a 2311 disk track.

Table 40.10.5. 1410/7010 CPU features supported and unsupported by the DOS 1410/7010 Emulator program

Supported	Unsupported
Main storage up to 80,000 positions (1410) and 100,000 positions (7010) Inverted Print Edit Priority Processing Processing Overlap Channels one through four Floating-Point Arithmetic Store and Restore Status	1401/1410 Compatibility Mode 7010 Diagnostic instruction, Branch on C Bit Diagnostic instruction, Branch if Tape Indicator J(I)K 7010 Memory Protect and Program Relocate 7010 Interval Timer

Table 40.10.6 1410/7010 devices not supported by the DOS 1410/7010 Emulator program

1311 Disk Storage Drive
1405 Disk Storage
Paper tape punches
Paper tape readers
Magnetic character readers
7340 Hypertape Drive
2321 Data Cell Drive
Teleprocessing devices

40:15 OS DOS EMULATOR PROGRAM

The availability of the OS DOS emulator offers DOS Version 3 and 4 (Releases 25, 26, and 27) users who upgrade to a Model 145 the opportunity to convert to an OS operating environment more easily than is possible without the use of emulation. In addition, the OS DOS emulator user can benefit from the use of integrated emulation, since the OS DOS emulator can execute concurrently with other OS jobs.

The OS DOS emulator for the Model 145, which is the same DOS emulator provided for System/370 Models 135 and 155, is a combination of the OS DOS emulator processing program and the standard OS/DOS Compatibility feature. This feature provides the relocation necessary for execution of a DOS supervisor and DOS programs under OS control in any processing program storage location. An OS MFT, MVT, VS1, or VS2 control program generated for a Model 145 is required also.

The OS DOS emulator program and the DOS system being emulated (DOS supervisor and up to three processing program partitions) execute together in an MFT minimum partition size of 39K (40K if storage protection is included in the MFT control program) or a minimum MVT region size of 46K. The OS DOS emulator program and tables require a minimum of 23K plus another 4K if I/O staging is used. Additional OS DOS emulator program storage may be required depending on the I/O devices used. Up to ten I/O devices are supported in 23K, and 250 bytes are required for each additional device. The I/O staging requirement of 4K supports unblocked reader, printer, and punch records and residence of the required QSAM routines in the OS DOS emulator partition or region. The use of certain other emulator options will also require an increase in the size of the OS DOS emulator program.

The DOS system being emulated can be 16K, 24K, or 32K and up, in 4K increments. The OS DOS emulator is scheduled to operate in the same manner as any other OS job, and one or more OS DOS emulator jobs can execute concurrently with OS jobs if enough I/O devices and processor storage are available. In addition, the Model 145 OS 1401/1440/1460 and 1410/7010 Emulator programs can execute concurrently with the OS DOS emulator if enough resources are present.

The user need not make any changes to the existing DOS supervisor, job control statements, tape files, or disk files in order to use the OS DOS emulator. Modification of existing DOS programs is required only for programs that contain features unsupported by the OS DOS emulator.

The major advantages of the OS DOS emulator are:

- Transition from a DOS to an OS operating environment is smoother. The conversion of DOS source programs, job control, and data files to OS formats can be done gradually for emulated DOS jobs.

- Dedicated emulation is not required, thus allowing the user to take advantage of OS facilities.

- OS data sets and OS-compatible DOS files can be accessed by emulated DOS programs (without modification of the DOS programs) and by DOS programs that have been converted to OS format.

- Direct access volumes can contain both DOS files and OS data sets so that direct access devices can be accessed by concurrently executing OS jobs and DOS emulator jobs.

- Model 145 OS DOS emulator users can continue to use most IBM-supplied application programs (Type II and program products) that operate under DOS but not under OS and do not use QTAM, by emulating them under OS.

- Total system throughput can be increased by operation of the OS DOS emulator in a multiprogramming OS environment and by using the staged I/O option of the OS DOS emulator. The latter permits emulated DOS programs to use the data transcription facilities of the OS reader interpreter and output writer to handle their unit record functions. Use of the staged I/O option of the OS DOS emulator also eliminates the necessity of dedicating unit record devices to DOS emulation.

All operating environments, control program facilities, I/O devices, and telecommunications terminals supported by DOS Versions 3 and 4 (Releases 25, 26, and 27) can be emulated, with the following exceptions:

- Autotest
- OLTEP (which does not produce meaningful results)
- Model-dependent functions such as CS/30, CS/40, and the DIAGNOSE instruction (1400 emulation can be handled using the Model 145 OS 1401/1440/1460 Emulator program)
- Emulation of emulators that operate under DOS, for example, Model 145 integrated 1400/7010 emulators that operate under DOS
- QTAM
- 1259, 1412, and 1419 Magnetic Character Readers
- 1287 and 1288 Optical Character Readers in document mode if response times are required for pocket selection
- 1445 Printer
- 2260 Display Station (local)
- 7770/7772 Audio Response Units
- Storage protection within the DOS system being emulated (among the DOS supervisor and partitions)
- DOS (and OS) direct access volumes having nonunique volume serial numbers online concurrently

In addition, executable DOS programs cannot be handled that:

- Rely on known timing relationships of the DOS system
- Depend on HALT I/O (except for BTAM programs), READ DIRECT, WRITE DIRECT, or DIAGNOSE instructions for their operation
- Depend on the PCI flag in a CCW (this flag is ignored by the emulator)
- Modify or use information in CCW's after the CCW list is initiated with a START I/O instruction and before the I/O operation terminates
- Initiate the same CCW list for an I/O operation on more than one I/O device concurrently
- Specify data chaining in a CCW list when the DOS file is read via the OS input stream or written via the OS output stream

- Specify a CCW when the sum of the data address and count points to a storage location outside the DOS system area (DOS supervisor and partition area)

While a pseudo interval timer is maintained at simulated DOS decimal location 80, accurate time of day is not guaranteed, because the timer is running only when the OS DOS emulator partition/region is executing, and a time lag occurs during the interval required to update the timer.

The following I/O devices and telecommunications terminals are supported by the OS DOS emulator:

- 1052 Printer-Keyboard (emulated on a 3210-1 or 3215)
- 1285, 1287, 1288 Optical Readers (the latter two not in document mode if response times are required for pocket selection)
- 1403, 1443, 3211 Printers
- 1404 Printer for continuous form operations only (emulated on a 1403 or 3211 Printer)
- 1442, 2520, 2540 Card Read Punches
- 2501 Card Reader
- 2311 Disk Storage Drive
- 2314 and 2319 Direct Access Storage Facilities
- 2321 Data Cell Drive
- 2400- and 3400-series magnetic tape units
- 2671 Paper Tape Reader
- 3505 Card Reader
- 3525 Card Punch
- 3330-series disk storage
- All the start/stop and binary synchronous terminals supported by BTAM except the 2260 Display Station (local) and 7770/7772 Audio Response Units

EMULATOR JOB SUBMISSION AND GENERAL OPERATION

DOS emulation is initiated as a single-step OS job via the input stream. An OS DOS emulator job can consist of one or more DOS jobs. The OS DOS emulator program, which must reside in SYS1.LINKLIB or a user job library, is specified in the EXEC job control statement included in the job control for the OS DOS emulator job. The following also must be identified in the job control statements for the OS DOS emulator job:

1. The DOS system residence device and operator console device

2. The location of the DOS IPL input stream

3. I/O assignments for the staging of DOS unit record I/O operations

4. All the I/O devices except the operator console device that will be used by the DOS supervisor and DOS programs that are emulated as part of this DOS emulator job

The DOS input streams for DOS partitions can be located in the OS input stream or on separate data sets.

When the DOS emulation job is initiated, the DOS emulator program is loaded into the OS DOS emulator partition/region. The DOS emulator program performs control block and table initialization and initiates an IPL from the DOS system residence volume. Once the DOS supervisor has been loaded and has established the DOS partitions, DOS job execution begins. DOS programs are loaded into the defined DOS partitions and emulated. Messages to the operator from the DOS emulator program are issued in standard OS format and include a unique identification to indicate that they are OS DOS emulator messages. If the MCS option is included in the OS control program, all DOS emulation messages can be routed to a specific console, and thus isolated.

The entire OS DOS emulator partition/region operates with a nonzero storage protect key to prevent it from interfering with the OS control program and other executing OS jobs. Therefore, the DOS emulator program, the DOS supervisor, and other DOS jobs in the OS DOS emulator partition/region are not protected from inadvertent modification by an executing DOS program.

I/O DEVICE STAGING AND SHARING

If enough Model 145 processor storage is available, I/O staging can be used to increase OS DOS emulator job throughput and reduce the number of devices that have to be dedicated to the DOS emulation partition or region. It allows DOS unit record files SYSRDR, SYSPCH, and SYSLST to be emulated on direct access devices, using the OS reader interpreter and output writer. DOS job control statements and/or card input to DOS programs to be emulated can be placed in the OS input stream and will be transcribed by the reader interpreter to SYSIN data sets on direct access devices. Thus, emulated DOS job steps will obtain their card input from OS SYSIN disk data sets. Output from emulated DOS programs can be placed in OS SYSOUT data sets on disk to be transcribed to the printer or punch by an output writer.

The following should be noted about the use of I/O staging. In OS, a job is not placed in the input queue, from which all jobs are scheduled, until the entire job (job control and input stream data for the job) has been read by the reader interpreter. Similarly, SYSOUT data sets produced during job step execution are not placed in the output queue for transcription by an output writer until job termination.

Thus, if all DOS jobs to be emulated are grouped together as a single OS DOS emulator job, DOS emulation cannot begin until all DOS jobs (and their input stream data) have been read by the reader interpreter, and none of the SYSOUT data sets from completed emulated DOS jobs can be transcribed until the OS DOS emulator job itself terminates (all DOS jobs processed). This negates one advantage of I/O staging, which is the overlapping of unit record input and output data transcription with processing.

Therefore, consideration should be given to grouping DOS jobs into two or more OS DOS emulator jobs that execute one after the other in the OS DOS emulator partition/region. In addition, if the output from a particular DOS job is desired immediately, it should not be staged (written to a SYSOUT data set). The use of multiple OS DOS emulator jobs, instead of one, in an OS DOS emulator partition/region offers an additional advantage in optimizing device usage, as discussed below.

When the OS input stream contains multiple OS DOS emulator jobs, the automatic DOS IPL facility can be used to eliminate the need for the operator to enter DOS IPL commands each time a DOS emulator job is initiated (which causes an IPL of the DOS system). Another emulator option provides the capability of requesting abbreviated initial prompt messages. Normally, as an operational aid, four prompt messages are given to the operator at the time emulation begins. When abbreviated initial prompt is requested, a single prompt message is given, instead of four. This facility can be used once the operator has become familar with the emulator initialization procedure.

The I/O devices used in emulation need not have the same address on the Model 145 as on the current system. However, in all but two situations, the same device types currently being used by DOS programs must be used when these DOS programs are emulated (assuming the DOS programs are not modified to use a new device and then reassembled). First, I/O staging can be used for unit record devices. Second, certain I/O devices can be emulated on an I/O device of the same type with a different device number, without modification of existing DOS programs, if the old and the new I/O device are command code compatible as follows:

Current Device	Emulation Device
2540 reader punch (read operations)	3505 reader
2540 reader punch (punch operations)	3525 punch
1403 printer	3211 printer
2400-series tape units	3420 tape units

I/O operations and I/O error recovery procedures for emulated DOS programs are handled by the OS control program, with one exception. DOS ERP's are used to handle I/O errors for terminals supported by BTAM. All I/O devices to be used by emulated DOS programs must be allocated to the DOS emulation partition or region when the OS DOS emulator job is begun. All non-direct access devices allocated to a DOS emulator partition are dedicated to DOS emulation and cannot be allocated to any other executing OS jobs while DOS emulation is in operation in the partition. Direct access devices allocated to a DOS emulator partition do not have to be dedicated to DOS emulation. Therefore, optionally, a direct access volume that contains both DOS files and OS data sets can be accessed concurrently by OS jobs and a DOS emulation job.

The appropriate DD job control statements must be included for each DOS emulation job step for which direct access device sharing is to be supported. When direct access devices are shared, space allocation on direct access volumes and VTOC updating are performed by OS, and DOS equivalent functions are bypassed. Thus, OS password protection is active and DOS files with security protection indicated in their labels will be password-protected.

Restrictions on the use of direct access device sharing are the following:

- DOS files must be opened (to cause the execution of OS space allocation routines).

- Only single volume DOS files with up to 16 extents are supported on shared direct access devices.

- DOS job steps that specify actual direct access extents for new files are not supported since space will be allocated according to OS rules and the DOS specification is effectively overriden.

- The first extent of a DOS file with split cylinders must be the split cylinder extent and all data sets sharing a split cylinder must be processed by the same job step.

- Programs that access a file on a shared volume must not use embedded seek commands within channel programs.

When direct access device sharing is used, emulator size is increased. Direct access devices can be accessed by concurrently executing DOS emulator jobs and also by DOS partitions being emulated in the same OS DOS emulator partition.

It is the user's responsibility to ensure that all online OS DOS emulator direct access volumes have unique volume serial numbers with respect to other DOS and OS direct access volumes online at the same time.

Consideration should be given to grouping DOS jobs into multiple OS DOS emulator jobs according to the types and total number of I/O devices required. This can reduce the number of non-direct access I/O devices that have to be dedicated to a DOS emulation partition or region at any given time, thereby making more devices available to other OS jobs.

A direct data set sharing facility is also supported by the DOS emulator. This feature facilitates the conversion of emulated DOS programs to OS format by enabling DOS programs to be converted one or two at a time instead of a total application at a time. The DOS emulator supports processing of OS data sets and OS-compatible DOS files by OS programs and emulated DOS programs as follows:

- Existing DOS SAM and DAM files can be processed by DOS programs that have been converted to OS format as well as by emulated DOS programs. Existing DOS SAM and DAM files need not be converted because they are compatible enough with OS format data sets to be processed by OS access methods (QSAM and BDAM). DOS programs converted to OS format and any newly written OS programs that are to access existing DOS SAM and DAM files must use only those access method facilities that are supported by both OS and DOS. (Minor restrictions to the processing of DOS SAM and DAM files by OS QSAM and BDAM are indicated in Emulating DOS on IBM System/370 under OS, GC26-3777).

- Existing DOS ISAM files that are converted to OS ISAM data sets can be processed by DOS programs that have been converted to OS format and also by emulated DOS programs. OS ISAM data sets and DOS ISAM files are not compatible. Therefore, to enable emulated DOS programs to process OS ISAM data sets, the emulator logically maps DOS ISAM macros to OS ISAM macros during processing. When the OS ISAM data set is created from the DOS ISAM file, only those ISAM data formats and organizational options supported by both OS and DOS can be used. Similarly, DOS ISAM programs converted to OS format must use only those facilities supported by both OS and DOS ISAM. (See Emulating DOS on IBM System/370 under OS for specific limitations for ISAM data set sharing.) Use of the ISAM data set sharing facility increases the DOS emulator size. Additional emulator partition storage is also required for the OS ISAM access method requirement and for buffers.

INSTALLATION OF THE OS DOS EMULATOR

The following are the major steps that a DOS user must take to install the OS DOS emulator on a Model 145:

- Data processing personnel--systems analysts and designers, programmers, operators, etc.--must be educated on OS.

- The installation must decide which level OS control program will be used, MFT or MVT, and which functions and options are to be included.

- The desired OS operating system must be generated using a release of OS that includes Model 145 support. The DOS emulator option must be requested. Installation-designed routines, such as nonstandard tape label processing, accounting, etc., must be written, as required, and added to the generated operating system.

- The DOS emulator program must be obtained (it is distributed separately from OS) and added to the generated operating system.

- DOS jobs that cannot be emulated must be converted to OS format. This involves source program changes, conversion of DOS job control statements to OS job control statements, and, depending on data organization, conversion of DOS files to OS data sets. The amount of reprogramming required depends on the source language being used. In general, high-level language programs require much less modification than Assembler Language programs.

- The volume serial numbers of all existing DOS direct access files must be inspected for duplicates, and unique serial numbers should be assigned where necessary. Volume serial numbers assigned to newly created DOS files or OS data sets should be unique as well.

- The OS job stream should be planned, and consideration should be given as to how OS DOS emulator jobs are to be scheduled and executed, as discussed previously in this subsection. Note also that the total storage size of the DOS system being emulated may be reduced. For example, if one DOS processing partition is devoted to teleprocessing, CS/30, or CS/40, which are not emulated by the OS DOS emulator, this DOS partition is no longer required and its storage can be made available for allocation to an OS partition or region.

- Optionally, the size of the emulated DOS system can also be reduced by the removal of functions that are now provided by OS. For example, DOS POWER can be removed from a DOS system, since data transcription can be handled by the OS reader interpreter and output writer. The model-dependent DOS MCRR routine can be removed from a DOS supervisor, as Model 145 MCH and CCH routines contained in the OS control program will be used for machine and channel error handling.

  Note that alterations affecting the DOS supervisor normally require a system generation to be performed. In addition, any change resulting in a different starting address for a DOS partition means that existing nonrelocatable DOS programs executing in that DOS partition must be link-edited relative to the new address.

Figure 40.15.1 illustrates a 256K Model 145 configuration that supports one OS processing partition (P2), a transient 44K reader interpreter (P2), a resident output writer (P0), and emulation of a 128K DOS system (P1) using the staged I/O option. The ISAM data set sharing facility is not used.

Figure 40.15.2 illustrates a 160K Model 145 configuration that supports one OS processing partition (P2), a transient 32K reader interpreter (P2), a resident output writer (P0), and emulation of a 64K DOS system (P1), of which only 48K need be emulated. The staged I/O option and ISAM data set sharing are not used.

The examples assume a minimum MFT control program for the Model 145, a minimum emulator program size, and a Model 145 configuration that requires no more than 32K of control storage.

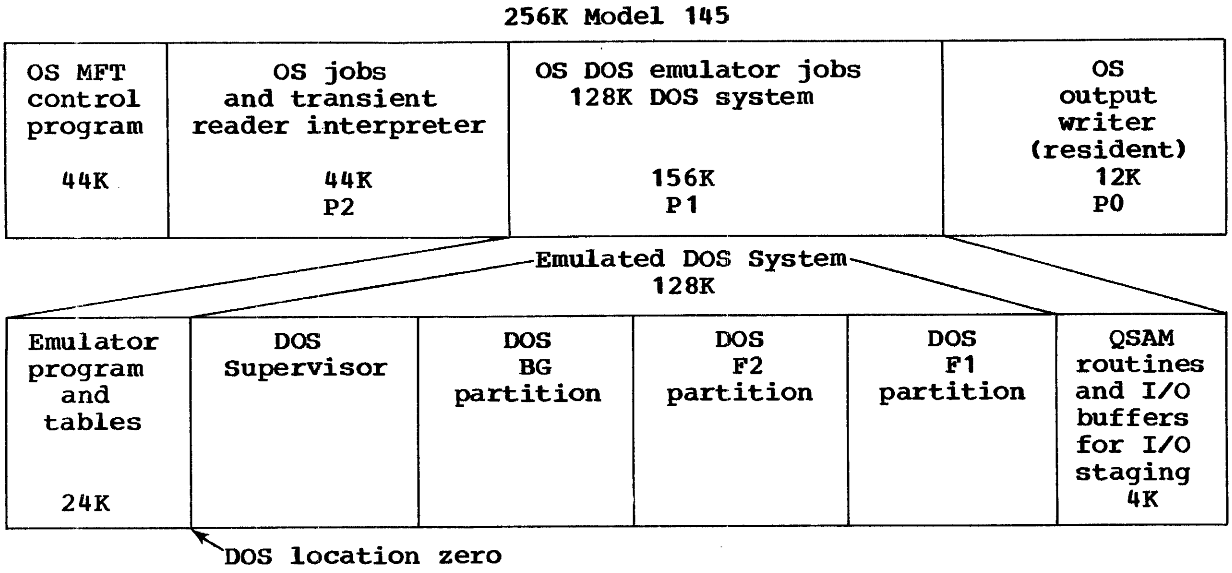

256K Model 145
OS MFT control program 44K
OS jobs and transient reader interpreter 44K P2
OS DOS emulator jobs 128K DOS system 156K P1
OS output writer (resident) 12K P0
Emulated DOS System 128K
Emulator program and tables 24K
DOS Supervisor
DOS BG partition
DOS F2 partition
DOS F1 partition
QSAM routines and I/O buffers for I/O staging 4K
DOS location zero

Figure 40.15.1. Sample 256K Model 145 configuration for emulation of a 128K DOS system

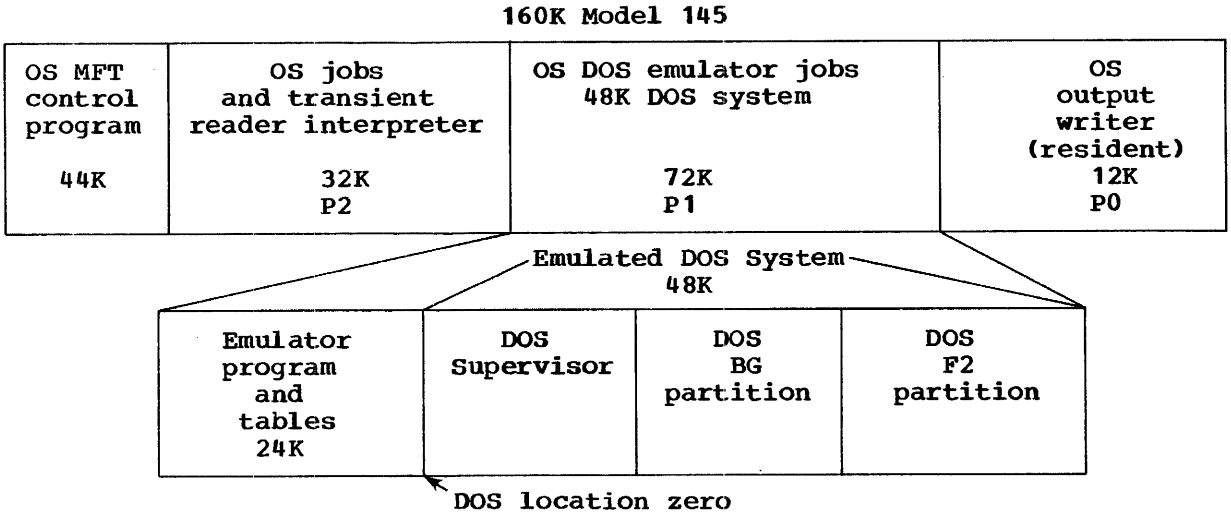

160K Model 145
OS MFT control program 44K
OS jobs and transient reader interpreter 32K P2
OS DOS emulator jobs 48K DOS system 72K P1
OS output writer (resident) 12K P0
Emulated DOS System 48K
Emulator program and tables 24K
DOS Supervisor
DOS BG partition
DOS F2 partition
DOS location zero

Figure 40.15.2. Sample 160K Model 145 configuration for emulation of a 48K DOS system

SECTION 50: RELIABILITY, AVAILABILITY, AND SERVICEABILITY (RAS) FEATURES

50:05 INTRODUCTION

With the growth of more and more online data processing activities, as distinguished from traditional batch accounting functions, the availability of a data processing system becomes a very essential factor in company operations, and complete system failure is extremely disruptive. Because of the growing frequency of online processing and the fact that the Model 145 is designed to operate in such an environment, IBM has provided an extensive group of advanced reliability, availability, and serviceability features for the System/370 Model 145. These RAS features are designed to improve the reliability of system hardware, to increase the availability of the computing system, and to improve the serviceability of system hardware components.

The objective of the RAS features of the System/370 Model 145 is to reduce the frequency and impact of system interruptions that are caused by hardware failure and necessitate a re-IPL. RAS features are as follows:

- Hardware reliability is enhanced through use of more reliable components.

- Recovery facilities, both hardware and program supported, not available for System/360 Models 30 and 40, are provided to reduce the number of failures that cause a complete system termination. This permits deferred maintenance.

- Repair procedures include more online diagnosis and repair of malfunctions concurrent with normal job execution in a multiprogramming environment in order to reduce the effect of such repairs on system unavailable time.

Each RAS feature, recovery or repair, is discussed in the remainder of this section.

The following recovery features are implemented in hardware:

- Retry of failing CPU operations

- ECC validity checking on processor and control storage to correct all single-bit and detect all double-bit errors

- I/O operation retry facilities, including channel retry data provided in the limited channel logout area, and channel/control unit command retry procedures to correct failing I/O operations

- Expanded machine check interruption facilities to facilitate better error recording and recovery procedures

The following recovery features are provided by programming systems:

- Recovery management support (RMS) to handle the expanded machine check interruption and channel retry data. MCH and CCH routines are provided for OS (except for PCP). MCAR and CCH routines are provided for DOS versions.

- Error recovery procedures (ERP) to retry failing I/O device and channel operations (OS and DOS)

- OBR and SDR routines (OS) and RMSR (DOS) to record statistics for I/O errors

- Environment recording, edit, and print program (EREP) for OS and DOS to format and print error log records

- I/O RMS routines (OS)--alternate path retry (APR) and dynamic device reconfiguration (DDR)--to provide additional recovery procedures after channel or I/O device failures

- Checkpoint/restart (OS and DOS) and warm start facilities (OS) to simplify and speed up system restart procedures after a failure necessitates a re-IPL

The following repair features are provided:

- Online Test Executive program (OLTEP) and Online Tests (OLT's) that execute under operating system control (OS and DOS) and provide online diagnosis of I/O device errors for most devices that attach to the Model 145

- Microdiagnostics to locate the malfunctioning field-replaceable unit

These aids are designed to enhance system availability. In many cases, the system can run in a degraded mode so that maintenance can be deferred to scheduled maintenance periods. When solid failures do occur, their impact can be reduced by faster isolation and repair of the malfunction than is currently possible.

The programmed recovery features discussed in this section are those for OS MFT and MFT and DOS Versions 3 and 4. The programmed recovery provided by the virtual storage operating systems is discussed in the optional programming systems supplements.

50:10 RECOVERY FEATURES

Additional hardware, which attempts correction of most hardware errors without programming assistance, has been included as part of the basic Model 145 system. The control program can be notified, via an interruption, of both intermittent and solid hardware errors so that error recording and recovery procedures can take place.

AUTOMATIC MICROINSTRUCTION RETRY

Detected CPU hardware errors can be retried automatically by microinstruction retry hardware. Retry can take place after an error occurs in any instruction, after failures that occur during interruption time when status information is being saved, after errors that occur during status saving for I/O instructions, etc. Even I/O instructions are retried automatically by the hardware without an intervening I/O interruption. (Either a machine check or an I/O interruption is taken, if the CPU is enabled for these interruptions, depending on the success of the I/O instruction retry and the point in the operation at which the error occurred.)

Microinstruction retry is accomplished by additional microprogram routines and hardware included in the Model 145. The failing CPU operation is retried by the microprogram up to eight times before it is determined that the error is uncorrectable. Checkpoints are taken and data is saved in backup locations during the operation of instructions that alter data as they execute, so that instructions can be retried from the point of correct execution.

When the CPU is enabled for machine check interruptions, an interruption takes place after a CPU error occurs and is retried. If microinstruction retry was successful, the failure need only be recorded; if unsuccessful, programmed recovery procedures are required.

The microinstruction retry feature provides the Model 145 with the ability to recover from intermittent CPU failures that would otherwise cause a system halt and necessitate a re-IPL or cause an executing program to be terminated. Corrected errors are logged by recovery routines for later diagnosis during scheduled maintenance periods, thereby increasing system availability.

Retry of failing CPU operations on the Model 40 is not provided by either system hardware or programming support.

ECC VALIDITY CHECKING ON PROCESSOR AND CONTROL STORAGE

The ECC method of validity checking on both processor and control storage provides automatic single-bit error detection and correction. It also detects all double-bit and some multiple-bit processor and control storage errors but does not correct them. Checking is handled on an eight-byte basis, using an eight-bit modified Hamming code, rather than on a single-byte basis, using a single parity bit. However, parity checking is still used to verify other data in a Model 145 system that is not contained in processor or control storage. Models 30 and 40 use parity checking for main storage data verification.

Data enters and leaves storage in the CPU through the storage adapter unit, which performs ECC validity checking on each doubleword. Another storage adapter is contained in the processor storage frame. When a doubleword (72 bits, as shown in Figure 50.10.1) is fetched from processor or control storage, the appropriate storage adapter unit checks the eight-bit ECC code to validate the 64 data bits. If the data is correct, the adapter unit generates the appropriate parity bit for each of the eight data bytes and reformats the doubleword to look as shown in Figure 50.10.2. If a single-bit error is detected, the identified data bit in error is corrected automatically by the corrector unit in the storage adapter and sent to the CPU. A corrected doubleword is sent back to control storage but not back to processor storage. When a doubleword is to be placed in processor storage by a program or in control storage during microprogram loading, the storage adapter unit strips the eight parity bits, constructs the necessary eight-bit ECC code, and appends the code to the 64 data bits. The 72 bits are then stored as shown in Figure 50.10.1. Additional CPU time is required to correct a single-bit error that occurs for a fetch to control storage.

When a single-bit storage error occurs, the hardware also determines whether the error is intermittent or solid by retrying the storage operation to see whether the error occurs again. With one exception, only intermittent single-bit storage errors can cause a machine check. When an intermittent single-bit storage error is detected and corrected during the execution of an instruction or I/O operation, a machine check pending latch is set on and the operation continues. At the completion of the CPU operation, a machine check interruption occurs to allow error recording to be done unless the CPU has been disabled for ECC correction interruptions. The occurrence of a machine check interruption after an intermittent single-bit processor or control storage correction is dependent on the setting of three ECC mode bits in a special mode register in the CPU and on a mask bit (recovery mask) in a control register. The mode register bits can be set by using the DIAGNOSE instruction.

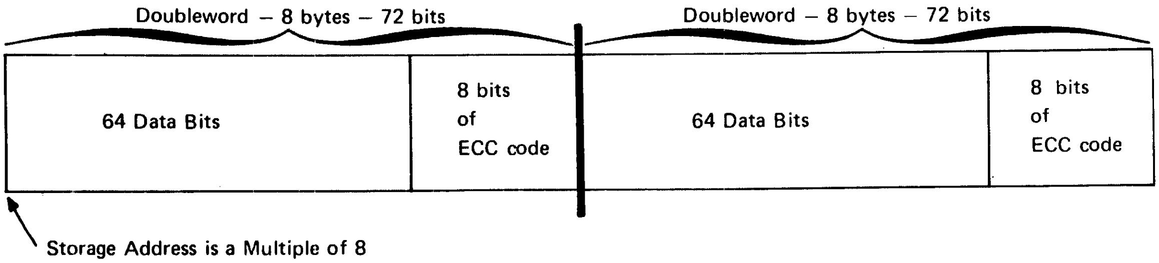

Doubleword – 8 bytes – 72 bits
Doubleword – 8 bytes – 72 bits
64 Data Bits
8 bits of ECC code
64 Data Bits
8 bits of ECC code
Storage Address is a Multiple of 8

Figure 50.10.1. Data representation used in Model 145 processor and control storage

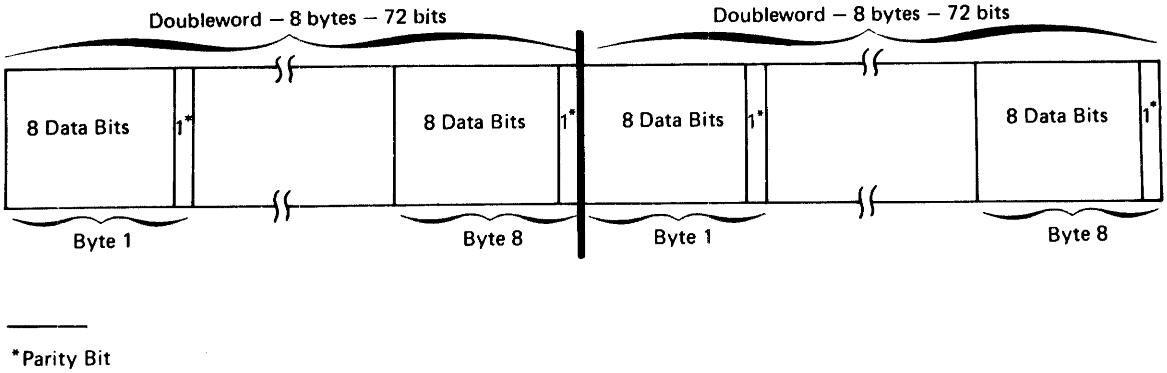


Figure 50.10.2. Data representation used in Models 30 and 40 processor storage and in the Model 145 in other than processor and control storage

One ECC mode bit controls machine check interruptions for intermittent processor storage single-bit corrections. It can be set to full recording mode to allow an interruption after each correction so that error logging can occur (if the control register mask bit is on also) or to quiet mode to disable the CPU for interruptions after single-bit processor storage corrections occur (without regard for the control register mask bit setting). The other two ECC mode bits control machine check interruptions for control storage single-bit corrections. Three modes are possible: full recording or quiet mode, as described for processor storage, and threshold mode. When threshold mode is in effect, a machine check interruption is taken after a certain number of single-bit control storage corrections have occurred in a given time interval. (These threshold values are hardwired.) When the threshold is exceeded, the hardware automatically establishes quiet mode for control storage single-bit corrections.

If a machine check interruption is taken after correction of a single-bit storage error, identification of the failing processor or control storage address and bit in error is provided in a fixed storage area (discussed in the machine check interruption explanation).

When a double- or multiple-bit storage error involving a CPU operation is detected, the microinstruction retry procedure is initiated. If a single-bit, instead of a double- or multiple-bit error results from one of the retries, the error is corrected and system operation continues as described above for a single-bit error. If the double- or multiple-bit error persists, a machine check interruption occurs and the error location is indicated in fixed storage. Error logging and recovery procedures should then be performed. When a double- or multiple-bit processor storage error occurs during an I/O operation, it is reported during the ensuing I/O interruption so that error recording and I/O retry procedures can be executed. A machine check interruption is not taken.

The ECC feature increases Model 145 system availability by permitting system operation to continue normally after single-bit processor and control storage errors occur and are corrected. Any processor storage errors on Models 30 and 40 necessitate at least termination of the processing program involved, since neither hardware nor programmed retry of processor storage errors is provided for these systems, nor is correction of control storage errors provided.

I/O OPERATION RETRY

Channel retry and command retry features are provided to reduce the number of abnormal program terminations and unscheduled system halts that occur because of I/O errors.

- Channel retry

  This feature has been implemented to ensure that most failing channel operations can be retried by error-handling routines. Both a limited and an extended channel logout are implemented. When a channel error or a CPU error associated with a channel operation occurs, the channel status word (CSW) and a new limited channel logout word are stored in the fixed lower storage area (I/O communications area) during the I/O interruption. The limited channel logout data provides additional, more exacting status information about the channel failure. The CCH routine passes this data to a device-dependent error recovery routine to be used in the retry of the failing I/O operation.

  Model 145 channels also attempt to log out any time a CSW is stored that indicates an interface control check or a channel control check error, if the CPU is enabled for I/O extended logouts. Up to 24 words are stored, beginning at the location indicated in the word at location 172, the I/O extended log pointer. This data is to be logged for later diagnosis by customer engineers.

  Channel error retry routines (channel check handlers) for System/360 models are provided only for Models 65 and higher. However, after a channel error occurs, these systems do not always present enough information to the error recovery routines to enable them to retry the failing operation. In other cases, the channel may be left in a condition in which retry is impossible after a channel malfunction. Model 145 hardware improvements eliminate these two situations in most instances.

- Command retry

  Command retry is a channel/control unit procedure that can cause an improperly executed command in a channel program to be retried automatically by hardware so that an I/O interruption and programmed error recovery are not required. An indication is presented when the control unit recognizes this situation. The byte multiplexer channel will not perform a command retry.

  The command retry feature is implemented in the control unit of 3330-series and 2305 disk storage and is discussed in Section 20.

EXPANDED MACHINE CHECK FACILITIES

Implementation of the machine check class of interruption on the System/370 Model 145 has been expanded in order to enhance error recording and error recovery procedures. Programming support of the extended machine check interruption is provided by the Model 145 MCH and MCAR routines of OS and DOS, respectively.

The machine check interruption facilities of the Model 145 differ from those of Models 30 and 40 as follows:

- Five types of machine check are defined.
- A machine check interruption occurs to permit the recording of errors corrected by the hardware as well as to allow recovery routines to handle errors that cannot be corrected by hardware.
- Machine check interruption masking is expanded to handle selective disabling and enabling of the CPU for the interruption types defined.
- The size of the fixed area in lower processor storage is increased to accommodate the storing of additional machine status and diagnostic information when a machine check interruption occurs.
- Error conditions are defined that cause the Model 145 to stop functioning immediately because the nature of the machine malfunction prevents valid processing from continuing.

The Model 145 presents one of five types of machine check interruption conditions, depending on the specific machine malfunction, and each type of interruption is maskable. Machine check interruption conditions are either <u>repressible</u> or <u>exigent</u> (formerly called <u>soft</u> or <u>hard</u>). A repressible machine check condition exists after the hardware has been successful in correcting an error or after an error has occurred that does not prevent continued successful execution of instructions. An interruption can occur after a repressible machine check condition so that the failure can be recorded. System operation continues after the error is logged. For example, if an error occurs during the execution of an instruction and if the microinstruction retry hardware corrects the error by reexecuting the failing instruction, a repressible machine check condition exists at the completion of the successful execution.

An exigent machine check condition exists when hardware retry fails or is not possible. Exigent conditions are those that prevent successful execution of the current instruction. For example, if the microinstruction retry hardware has not corrected a failing instruction after eight retries, an exigent rather than a repressible machine check condition exists after the last unsuccessful retry.

Figure 50.10.3 shows the layout of the model-dependent fixed processor storage in the Model 145. The total fixed storage area consists of four areas: the fixed locations in decimal addresses 0-159 (shown in Figure 10.10.2), the I/O communications area in locations 160-191 (used during EC mode I/O interruptions), the fixed logout area in locations 216-511, and the CPU extended logout area of 192 bytes, which begins at location 512 and continues to location 703 unless the logout pointer is changed by programming. The layout of the Model 145 fixed locations from address 160 to 703 is the same in BC and EC modes. The I/O address is stored in locations 185 to 187 only in EC mode, however.

A logout to the System/370 model-dependent fixed logout area (216-511) occurs when any type of machine check interruption is taken. The data stored is processed by recovery management routines. The fixed logout area data indicates the reason for the interruption in the machine check code and the region code. The values of the CPU timer and clock comparator are stored in locations 216 and 224, respectively, wherever a machine check occurs, if this feature is present in the system. The save areas in the logout area preserve the status of the system at the time of the machine check interruption and contain the contents of the general, the floating-point, and the control registers.

	160	Reserved				I/O COMMUNICATIONS AREA 160 – 191 *Stored for EC mode operations only
	168	Channel ID		172 I/O extended logout pointer		
	176	Limited channel logout		180 0		
	184	0	* I/O address	188 0		
	192	Unused				
	216	Contents of CPU timer				FIXED LOGOUT AREA 216 – 511 • Layout varies by System/370 model • Always logged on a machine check interruption • Processed by RMS
	224	Contents of clock comparator				
	232	Machine check code				
	240	Unused				
Region code 252-255	248	Failing processor storage address		252 Failing control storage address	Bit corrected	Type of storage error
	256	Unused				
	352	Floating point register save area				
	384	General register save area				
	448	Control register save area				
	512	CPU extended logout area — 192 bytes (Pointer in control register 15 set to 512 at IPL)				CPU EXTENDED LOGOUT AREA • Model dependent • Logged on all machine checks and first and eighth micro-instruction retry if not disabled

Figure 50.10.3. Model 145 model-dependent fixed storage locations

The model-dependent CPU extended logout area begins at the address specified in control register 15, which is set by the hardware to decimal location 512 by an IPL or on a system reset. The length of this extended logout area on the Model 145 is 192 bytes. Currently, however, only 112 bytes of this area are used, and this length instead of 192 is stored in the machine check code. A logout to the extended area occurs for all types of machine check interruptions if the log suppression mask bit in control register 14 is a one. This data can be recorded by recovery management routines.

Figure 50.10.4 illustrates the layout and the contents of the eight-byte machine check code stored in processor storage locations 232-239. The machine check code indicates which type of interruption occurred, the validity of certain fields stored in the fixed logout area, and the length of the stored CPU extended logout area. The four-byte region code is used to further describe processor and control storage errors. The region code identifies an interruption caused by a double-bit control storage error, a control or processor storage intermittent failure, or the exceeding of a threshold value for single-bit control storage errors.

Table 50.10.1 lists the machine check types defined for the Model 145. They are described in the discussion that follows. The mask bits used to enable or disable the CPU for interruptions for each type are indicated and the setting of the machine check and region codes are discussed. PSW bit 13 and two other mask bits are used to enable and disable the CPU for machine check interruptions. The recovery mask (R) and external mask (E) bits are contained in control register 14 and operate subject to PSW bit 13. If PSW bit 13 is a zero, the CPU is disabled for all machine checks. If PSW bit 13 is a one, the settings of the two additional mask bits determine whether or not interruptions, other than Instruction Processing Damage and System Damage, will be taken (refer to Figure 50.10.4).

Table 50.10.1. Model 145 machine check interruptions

Mask Bit(s)	Interruption Type and Cause	Machine Check Condition
PSW 13 and R	System Recovery •CPU error corrected by retry •Intermittent single-bit processor or control storage error corrected	Repressible
PSW 13 and E	Interval Timer Damage	Repressible
PSW 13 and E	Timing Facilities Damage •Time of day clock •Clock comparator •CPU timer	Repressible
PSW 13	System Damage •Irreparable hardware malfunction	Exigent
PSW 13	Instruction Processing Damage One of the following occurs during instruction execution: •Unretryable CPU error •Uncorrectable CPU error •Double-bit processor or control storage error •Storage protect key failure	Exigent

Repressible Machine Check Interruptions

Repressible machine check interruptions are as follows:

- System Recovery. This interruption occurs if both PSW bit 13 and the recovery mask bit are on. It is caused by a successful microinstruction retry or a single-bit intermittent processor or control storage error correction.

  The SR bit in the stored machine check code (bit 2) is used to indicate successful hardware recovery. When the SR bit is on without another recovery bit, an error has occurred in the normal functioning of the CPU and the CPU operation has been retried successfully by microinstruction retry hardware. CPU extended logout data is generated after the first and eighth retry.

Fixed Logout Area Locations 232–239

	0-8 Machine Check Types										14-15 Machine Check Tense		16-18 Storage Error			20-31, 46, 47 Validity Bits																48-63 CPU Extended Log Length
	SD	PD	SR	TD	CD	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	BACKED UP	DELAYED	SE	SC	KE	UNUSED													UNUSED			Zero if no logout or 112 bytes
Bit	0	1	2	3	4	5	6	7	8	9-13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32-45	46	47	48 … 63

Bit	Interruption Type
0	SD – System Damage
1	PD – Instruction Processing Damage
2	SR – System Recovery
3	TD – Timer Damage
4	CD – Timing Facilities Damage

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 occurred)
46	CPU Timer Value
47	Clock Comparator Value

Figure 50.10.4. Model 145 machine check code

The SC bit in the stored machine check code (bit 17) is used together with the SR bit to indicate that the ECC hardware corrected a single-bit intermittent processor storage error and passed correct information about the error to the CPU. The failing processor storage address is stored in locations 248-251. The region code indicates the bit corrected and whether or not the error was intermittent. When a control storage error occurs, the failing control storage address, the bit corrected, and whether the error was intermittent are indicated in the region code. When the interruption occurs because the single-bit control storage threshold has been exceeded, this fact is also indicated in the region code. Validity bits 24 and 25 indicate that the failing storage address and region code, respectively, have been stored correctly. Only error recording is required for this interruption.

- Interval Timer Damage. This interruption occurs if PSW bit 13 and the external mask bit are on. It indicates damage to the timer. Programmed validation procedures and error logging are required.

- Timing Facilities Damage. This interruption occurs if both PSW bit 13 and the external mask bit are on. The CD bit in the stored machine check code (bit 4) is used to indicate that an error occurred in the time of day clock that renders the clock invalid. Once this invalid indication has been given, subsequent STORE CLOCK instructions cause the condition code in the current PSW to indicate the fact that the clock is invalid. Error logging is required as a result of clock failure. This interruption also occurs if a STORE CLOCK COMPARATOR or a STORE CPU TIMER instruction is issued and the addressed timing feature has an error condition.

Exigent Machine Check Interruptions

Exigent machine check interruptions are as follows:

- Instruction Processing Damage. This interruption occurs if PSW bit 13 is on. The PD bit in the stored machine check interruption code (bit 1) is used to indicate that an error occurred during the execution of the instruction indicated by the machine check old PSW. The error was either a double-bit processor or control storage failure, a storage protect key failure, or a CPU error that was unretryable or that could not be corrected by microinstruction retry hardware.

  If a double-bit processor or control storage failure caused the interruption, the address of the failing storage area is indicated in locations 248-251 or 252-253, respectively. The SE (storage uncorrected) bit is turned on for a processor storage error. Byte 3 of the region code indicates a control storage error. The KE bit in the machine check code (bit 18) is set on when a storage protection failure occurs. The processor storage block affected is indicated in the failing storage address field (248-251).

  If an unsuccessfully retried CPU failure caused the interruption, the backed-up bit in the machine check code (bit 14) indicates the extent of the damage that occurred, if any. If the backed-up bit is on, it indicates that no source data has been changed and that the PSW registers and storage reflect the valid state that existed at the beginning of the instruction. (For a CPU error, an extended logout is generated on the first and eighth unsuccessful retry.)

Error logging and the execution of recovery procedures are required after this type of interruption.

- System Damage. This interruption occurs if PSW bit 13 is on. The SD bit in the stored machine check code (bit 0) is used to indicate that an irreparable CPU failure occurred that was not a result of the execution of the instruction indicated in the machine check old PSW. An unsuccessful retry of an interruption or a wait state, control register damage, etc., are examples of system damage errors. System damage also is indicated if the error cannot be identified as one of the other types of machine check interruption. Programmed error recovery is not possible after this type of failure.

<u>Modes of System Operation for Machine Check Interruptions</u>

Two modes of system operation for machine check interruptions are possible: <u>full recording</u> mode and <u>quiet</u>, or nonrecording, mode. In full recording mode the CPU is enabled for all machine check interruption types and all types cause an interruption to be taken and logouts to occur except for intermittent single-bit storage error corrections. This is the normal mode of Model 145 operation. In quiet mode, the CPU is enabled for all or certain machine check interruptions for repressible conditions. Quiet mode can be used to permit system operation without error recording for all or certain repressible conditions when a large number of transient (correctable) errors are occurring. It also can be used to allow Model 145 operation under the control of an operating system without Model 145 machine check handling routines included.

A check-stop status and a check-stop-control bit have been defined for the Model 145 (formerly called hard stop status and the hard stop bit). The check-stop-control bit is located in control register 14 with the other two mask bits discussed. If a check-stop condition occurs, the Model 145 system ceases all operations immediately without the occurrence of a logout to the fixed area. Check-stop is initiated by hardware rather than by programming.

Generally speaking, a check-stop situation is caused by the occurrence of an exigent machine check condition during the processing of a previous exigent machine check condition. Implementation of a check-stop status prevents system operations from continuing when the nature of the machine malfunction prevents the system from presenting meaningful status data.

The state of the Model 145 after IPL or a system reset is:

1. Recovery reports are not specified. The processor storage single-bit ECC correction mode bit is set to quiet mode and the control storage ECC mode bit is set to threshold mode for single-bit corrections. Thus, successful microinstruction retries and single-bit ECC corrections on both control and processor storage do not cause machine check interruptions.

2. External interruptions are specified. Interval Timer Damage or Timing Facilities Damage causes a machine check interruption.

3. PSW bit 13 normally is set to one by the IPL PSW (it is set to zero by the IPL system reset procedure) to specify System Damage and Instruction Processing Damage interruptions. Therefore, an irreparable system error, an unretryable CPU failure, an unsuccessfully retried CPU failure, or a double-bit processor or control storage error associated with instruction processing causes an exigent machine check interruption.

4. The check-stop-control bit is on.

5. CPU extended logout is specified and control register 15 points to location 512 as the beginning of the CPU extended logout area.

6. I/O extended logouts are specified.

These settings cause the Model 145 to run in quiet mode for hardware-corrected errors. If the Model 145 is to operate in full recording mode, the appropriate mask bits must be altered by the control program.

MACHINE CHECKS ON SYSTEM/360 MODELS 30 AND 40

A machine check situation in Models 30 and 40 results from hardware detection of a machine malfunction or of a parity error. Bad parity can occur in main storage, in local storage, in a register, in an adder, etc. Error correction is not attempted by Model 30 or 40 hardware when a machine check occurs (except for some instruction retry capability in the Model 30). If the machine check mask in the current PSW (bit 13) is enabled, a machine check causes an interruption and a diagnostic scan-out occurs, starting at location 128. The number of bytes logged is model dependent.

If an OS MFT or MVT SER routine for the Model 40 or the DOS MCRR routine for the Model 30 or 40 is present, it gains CPU control after a machine check interruption, and the error is logged. A retry of the failing operation is not provided by these routines and the affected program is terminated abnormally. If a recovery routine is not present, the system is placed in a wait state when a machine check interruption occurs. (An OS control program for a Model 40 must contain a machine check handling routine, SER0 or SER1, as of Release 17.)

RECOVERY MANAGEMENT SUPPORT (RMS) FOR OS MFT AND MVT

RMS for the Model 145 consists of extensions to the facilities offered by RMS routines currently provided for Models 65 and up. The two RMS routines, machine check handler (MCH) to handle machine check interruptions and channel check handler (CCH) to handle certain channel errors, are included automatically in MFT and MVT control programs generated for the Model 145.

The two primary objectives of RMS are (1) to reduce the number of system terminations that result from machine malfunctions and (2) to minimize the impact of such incidents. These objectives are accomplished by programmed recovery to allow system operations to continue whenever possible and by the recording of system status for both transient (corrected) and permanent (uncorrected) hardware errors.

Machine Check Handler

After IPL of a control program containing Model 145 RMS routines, the recovery mask bit is on to permit recording of microinstruction retry corrections, quiet mode is established for single-bit processor storage corrections, threshold mode is established for single-bit control storage corrections, and external interruptions are specified as are CPU and I/O extended logouts. MCH receives control after the occurrence of machine check interruptions for both repressible and exigent conditions.

Repressible Machine Checks. When a System Recovery machine check occurs to indicate a successful microinstruction retry, MCH formats a recovery report record to be written in the system error recording data set SYS1.LOGREC. This record contains pertinent information about the error, including the data in the fixed logout area, an indication of the recovery that occurred, identification of the job, job step, and program involved in the error, the date, and the time of day. MCH schedules the writing of the recovery report record and informs the operator that a machine check has occurred.

MCH supports an operator MODE command that can be used to enable the CPU for interruptions after intermittent single-bit error corrections. The operator can establish full recording mode for intermittent single-bit processor and/or control storage ECC corrections. These corrected errors will then be recorded and the operator will be notified. The operator can also reestablish quiet mode for processor storage corrections and threshold mode for control storage corrections. (The operator is notified when a switch from threshold to quiet mode is made for control storage corrections.) In addition, the operator can establish quiet mode for control storage corrections.

A capability for the operator to switch to quiet mode for successful microinstruction retries is not provided, as is discussed for the DOS MCAR routine; thus, recording always occurs for these errors.

The operator also is informed of the occurrence of a Timing Facilities Damage or an Interval Timer Damage machine check interruption. Error recording is performed, after which the system is placed in a wait state if a timing facilities error occurred. System operation continues after an interval timer error. (See Figure 50.10.5 for the general flow of OS MCH processing after repressible machine check conditions.)

Exigent Machine Checks. When an Instruction Processing Damage machine check occurs (uncorrectable or unretryable CPU error, double-bit processor or control storage error, or storage protect key failure), MCH assesses the damage.

MCH attempts to identify the task associated with the error so that the task can be terminated abnormally. A damage report record, which contains both the fixed and CPU extended logout area da+a, the recovery action taken, the program identification, the date, and the time of day, is prepared and logged. System operation continues if the error task associated with an uncorrectable error can be identified and terminated. System operation halts and a re-IPL is required if the error damages a portion of the control program, or if the error cannot be associated with a specific task. The operator is informed of whatever action is taken.

When a System Damage machine check occurs, programmed recovery is not possible, and MCH places the system in a wait state after a logout and termination procedures are attempted.

The MFT and MVT MCH routine for the Model 145 contains model-dependent routines and will not execute correctly on System/360 models or another System/370 model in BC mode. See Section 60:20 for a discussion of operating system portability. (For a discussion of system operation with an operating system without Model 145 RMS, see Section 60:35.)

Channel Check Handler

CCH receives control after a channel error causes an I/O interruption. CCH formats both an error information block (containing the limited channel logout data) for use by an ERP routine and a CCH error record for recording in SYS1.LOGREC. The latter contains status information from the logout area, the limited channel logout area, program identification, date, and time of day.

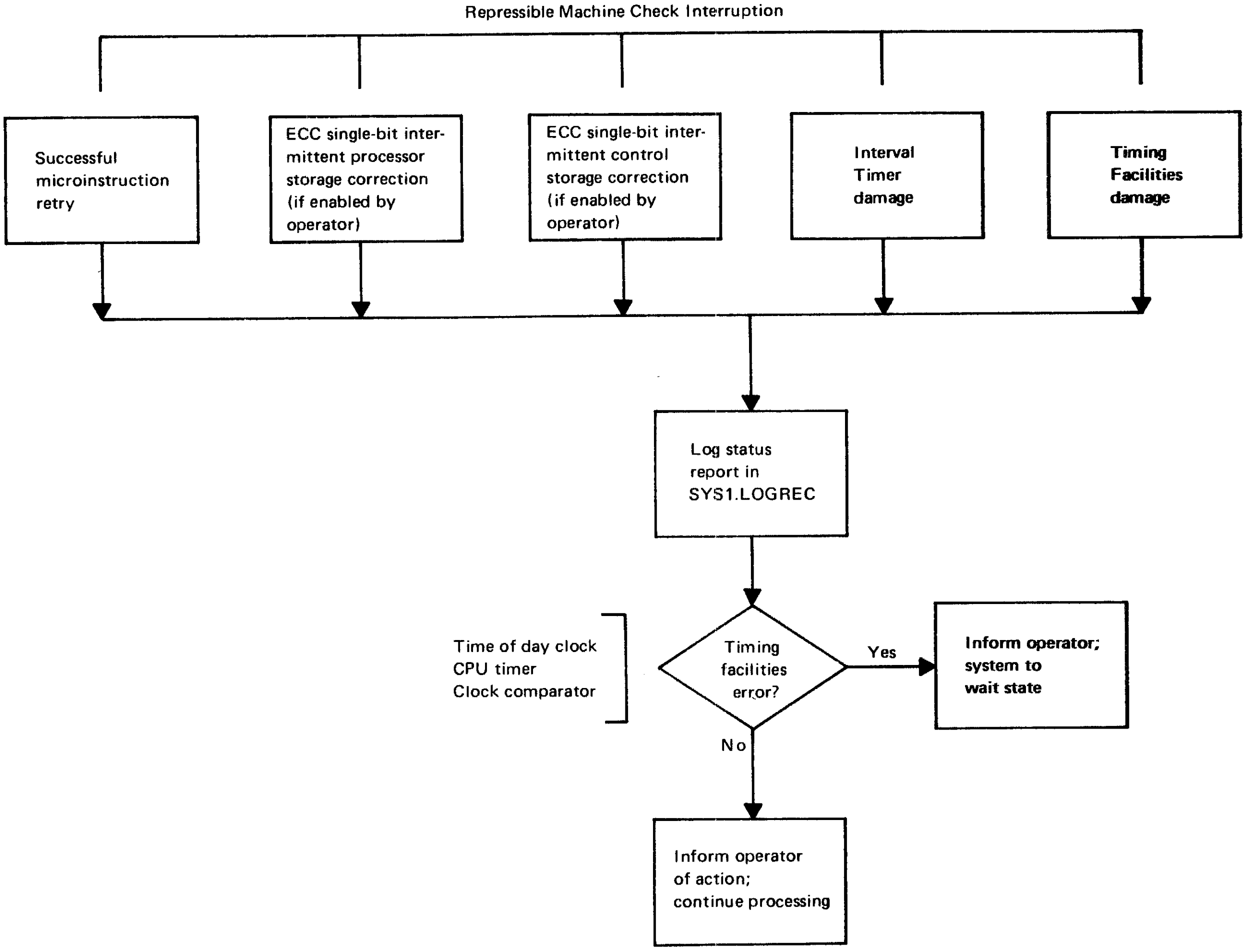

Repressible Machine Check Interruption
Successful microinstruction retry
ECC single-bit intermittent processor storage correction (if enabled by operator)
ECC single-bit intermittent control storage correction (if enabled by operator)
Interval Timer damage
Timing Facilities damage
Log status report in SYS1.LOGREC
Time of day clock
CPU timer
Clock comparator
Timing facilities error?
Yes
Inform operator; system to wait state
No
Inform operator of action; continue processing

Figure 50.10.5. General flow of OS MCH processing after repressible machine check conditions

If CCH determines that operating system integrity has been impaired by the channel error, control is given to MCH for error recording, and system operations are terminated. Otherwise, the error information block and error record are passed to the appropriate device-dependent error recovery procedure (ERP), which causes the error record to be logged and retries the failing I/O operation, using status information from the error information block. If a successful retry occurs, system operation continues. If the error is deemed permanent (uncorrectable), another error record is prepared and recorded by the outboard recorder routine (OBR), and the task involved is abnormally terminated (unless I/O RMS or a user-written permanent error handling routine is present). The operator is informed of the abnormal termination and system operation continues.

The CCH routine is structured in a manner that makes it model independent. A channel/model-independent module resides in the operating system nucleus. The required channel-dependent modules for the Model 145 included in the operating system at system generation time are loaded during the IPL procedure. The nucleus initialization program (NIP), using channel configuration data specified by the user at system generation time and the new STORE CHANNEL ID instruction, determines the types of channels present in the system. OS MFT and MVT CCH routines are, therefore, compatible for System/370 models operating in BC mode, for System/360 Models 65 and up, and for MP/65 systems.

Figure 50.10.6 at the end of this subsection shows the general flow of programmed error recovery procedures after an I/O interruption.

ERROR RECOVERY PROCEDURES (ERP'S) FOR OS

These device-dependent error routines are a standard part of the control program generated for any OS environment. OS ERP's are modified to accept and use limited channel logout data formatted by the CCH routine after a channel error.

When a channel or an I/O device error occurs on a Model 145, the appropriate ERP is scheduled to perform recovery procedures. If the error is corrected, operations continue normally. If the error cannot be corrected (it is permanent), error recording occurs. If I/O RMS or a user-written permanent error handling routine is not present, the affected task is abnormally terminated. The operator is notified of permanent I/O errors. (See Figure 50.10.6.)

STATISTICAL DATA RECORDER (SDR) AND OUTBOARD RECORDER (OBR) FOR OS

OBR and SDR routines are included in all OS control programs. These routines are requested by the ERP routines during their processing of error conditions. The SDR routine is requested when one of the error statistics counters becomes full. Counters are maintained in the resident control program storage area for each I/O device in the system configuration. SDR records error statistics in the appropriate SDR summary record for a device that is contained in the error log data set (SYS1.LOGREC). This ensures recording of temporary I/O device error data. The OBR routine records both temporary and permanent channel errors and writes an outboard record containing pertinent status data whenever a permanent error occurs for a device. SDR is also executed to write accumulated statistics for that device when a permanent error occurs. (See Figure 50.10.6.)

ENVIRONMENT RECORDING, EDIT, AND PRINT PROGRAM (EREP) FOR OS

OS EREP is a standard system utility that can be initiated as a job step via standard job control statements at any time. It is designed to edit and print the error records written by System/370 OS RMS routines. EREP contains model-dependent routines and, therefore, an EREP program that handles error records for one System/370 model in BC mode cannot be used to handle the error records for another System/370 model. It performs the following:

1. Edits and prints all error records contained in SYS1.LOGREC. These records have been constructed and/or written by MCH, CCH, OBR, and SDR routines.

2. Accumulates a history of specified record types from SYS1.LOGREC by creating or updating an accumulation data set

3. Edits and prints a summary of selected records from SYS1.LOGREC or an accumulation data set

I/O RMS FOR OS

I/O RMS routines are optional, model-independent routines in MFT and MVT environments. These reconfiguration procedures attempt to minimize the number of abnormal job terminations and unscheduled system halts that occur because of errors on channels or I/O devices.

The alternate path retry (APR) routine provides for the retry of a failing I/O operation on another channel path to the device involved, if one is available, when an uncorrectable channel error occurs. Thus APR, if present, is entered from a device-dependent ERP when a permanent error is deemed to exist after retry procedures have been attempted. If the I/O error is corrected using the alternate channel path, operations continue. If a permanent error still exists, the task is abnormally terminated unless the dynamic device reconfiguration routine is present. A malfunctioning channel path can be varied offline by the operator if necessary.

The dynamic device reconfiguration (DDR) routine permits the operator to move a demountable volume from one device to another of the same type when a permanent hardware error occurs and provides repositioning of the volume so that the failing I/O operation can be retried. A volume can also be demounted so that device cleaning procedures can be performed, and it can then be remounted on the same device. The DDR option also supports demountable system residence devices and unit record equipment. DDR is entered from a device-dependent ERP after a permanent channel or device error occurs on a demountable device. Task termination occurs if the error cannot be corrected and a user-written permanent error handling routine is not present. (See Figure 50.10.6.)

I/O RMS is not included in DOS support, which handles alternate channel paths only for tape unit switching and does not provide dynamic I/O device allocation by the supervisor.

ADVANCED CHECKPOINT/RESTART AND WARM START FACILITIES FOR OS

If the RMS and I/O RMS routines fail in their attempt to correct a hardware error and the error is one that causes program or system termination, the automatically provided advanced checkpoint/restart and warm start facilities of OS can be employed to minimize the impact of the termination on system operation. The automatic restart facility can be used to cause terminated programs to be rescheduled immediately without resubmission of their job control, so that a minimum of operator intervention is required. The operator must authorize all automatic job step restarts. If a permanent I/O device or channel failure caused the program termination, the device or channel can be varied offline. This will ensure allocation of a different device when the program step is reinitiated.

The warm start facilities of the control program provide automatic saving of SYSIN and SYSOUT data sets and input and output work queues so that processed work is not lost when a system termination occurs. The operator is informed of the status of jobs in execution when the system terminated, and these jobs should be restarted automatically from the beginning or from a checkpoint if the type of processing involved permits such a restart. System design should include planned restart procedures for unscheduled terminations of individual programs and the system. (See Figure 50.10.6.)

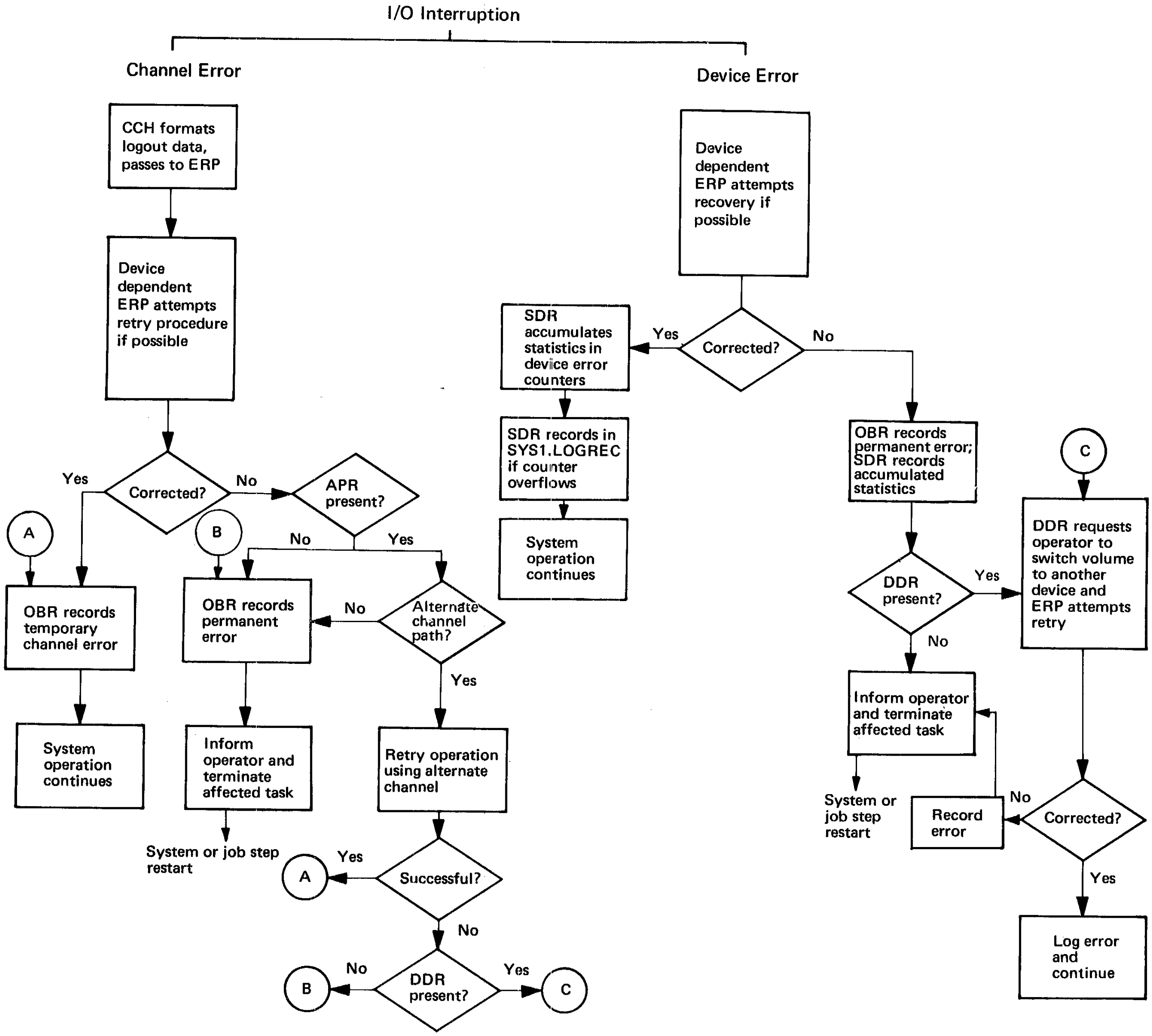

I/O Interruption
Channel Error
Device Error
CCH formats logout data, passes to ERP
Device dependent ERP attempts retry procedure if possible
Corrected?
Yes
No
APR present?
A
B
No
Yes
OBR records temporary channel error
OBR records permanent error
No
Alternate channel path?
Yes
System operation continues
Inform operator and terminate affected task
Retry operation using alternate channel
System or job step restart
A
Yes
Successful?
No
B
No
DDR present?
Yes
C
Device dependent ERP attempts recovery if possible
SDR accumulates statistics in device error counters
Yes
Corrected?
No
SDR records in SYS1.LOGREC if counter overflows
System operation continues
OBR records permanent error; SDR records accumulated statistics
DDR present?
Yes
No
Inform operator and terminate affected task
System or job step restart
C
DDR requests operator to switch volume to another device and ERP attempts retry
Record error
No
Corrected?
Yes
Log error and continue

Figure 50.10.6. General flow of OS error recovery procedures after an I/O interruption

RECOVERY MANAGEMENT SUPPORT (RMS) FOR DOS VERSION 4

Machine check analysis and recording (MCAR) and channel check handler (CCH) routines for the Model 145 represent an extension of the recovery support provided by DOS for System/360 Models 30, 40, and 50. Machine check and channel error recording is offered as an option for these models by the Machine Check Recording and Recovery (MCRR) routine.

MCAR, CCH, and the error recording routine RMSR (recovery management support recorder) are included automatically in any DOS Version 4 supervisor generated for a Model 145. RMSR, a new feature of DOS Version 4, performs all error recording functions. It replaces the following functions provided in DOS Version 3: outboard recorder (OBR), statistical data recorder (SDR), and the recording functions of MCAR, CCH, tape error by volume (TEBV), and error volume analysis (EVA).

Machine Check Analysis and Recording

After IPL of a control program containing Model 145 RMS routines, mask bits are set to ones and control register values are set to permit machine check interruptions and logouts to occur as discussed for OS MCH.

When a System Recovery machine check occurs to indicate a successful microinstruction retry, an environment record (recovery report), is constructed by MCAR. The environment record, which contains pertinent status information from the fixed area, recovery action, program identification, date, and time of day, is then passed by MCAR to RMSR to be written in the recorder file, whose symbolic unit name is SYSREC (corresponding to the SYS1.LOGREC recording data set of OS). The operator is informed that a repressible machine check has occurred.

If a repressible error condition occurs during execution of nonprivileged code (a processing program, for example), error recording takes place when the interruption occurs. If the repressible error condition occurs during execution of privileged code (supervisor code, for example), the CPU is disabled for machine checks due to repressible conditions, an indication that MCAR processing is required is stored, and control is returned to the interrupted privileged code. When execution of the privileged code terminates, control is given to MCAR to record the repressible error condition. Machine check statistics are lost during the interval between occurrence of the repressible error and its recording if another machine check condition arises. Specifically, a machine check interruption for an exigent condition during this interval will overlay the machine check logout for the repressible error condition, and any additional repressible machine check conditions will result in the loss of all but one of the repressible errors, since the Model 145 can keep only one machine check pending at a time for each machine check type.

Prior to relinquishing CPU control, MCAR determines whether or not an automatic mode switch from recording mode to quiet (nonrecording) mode should be made for microinstruction retry recoveries. Quiet mode is established by MCAR (the System Recovery mask bit is turned off) if the number of microinstruction retry corrections that occur during a given time interval exceeds the established error count threshold value for these corrections. (The time of day clock is used by MCAR to determine whether or not the time interval threshold has been exceeded.) The IBM-supplied threshold values can be altered during system generation or by the operator MODE command during system operation. The operator is informed of any mode switch made by MCAR and can switch back to recording mode any time thereafter. Quiet mode can be used to prevent SYSREC from being filled with CPU recovery reports when a large number of transient errors are occurring.

As described for the OS MCH routine, MCAR also supports an operator MODE command to permit the operator to enable the CPU for interruptions after single-bit intermittent processor and control storage corrections so that these errors can be logged.

If an error damages the time of day clock, MCAR places the system in quiet mode for microinstruction retry and ECC corrections, a message is issued to the operator, and processing continues. If the interval timer is damaged, a message is issued and processing continues.

When an Instruction Processing Damage machine check occurs (uncorrectable or unretryable CPU error, double-bit storage error, or storage protect key error) during the execution of supervisor (or any privileged) code, the system is placed in a check-stop state after an attempt is made to prepare and record a damage report record. MCAR does not attempt to refresh damaged supervisor code. The system is also placed in the wait state if an uncorrectable error occurs during an attempt to access critical data or programs contained on the SYSRES device. The occurrence of an Instruction Processing Damage interruption during processing program execution always results in termination of the task involved, after error recording occurs, and system operation continues.

MCAR performs repair procedures if a storage protect key failure or double-bit processor storage error occurs in a processing program partition. Validation of damaged processor storage is attempted by moving a doubleword of binary zeros and then ones into the area. If the storage protect key repair or storage validation procedure fails, dynamic reallocation of the partition is attempted. A routine is executed to determine in which half of the processing partition the storage error occurred. If the failure occurred in the high-order half of the partition, the upper boundary of the partition is lowered to the first 2K boundary address below the address of the error location. If the failure occurred in the low-order half of a foreground partition, the lower boundary of the foreground partition is raised to the first 2K boundary address above the address of the error location. (The lower boundary of the background partition cannot be raised.)

A message is issued to the operator to inform him of the new upper or lower boundary address and the current size of the partition whenever a successful reallocation is performed. Only programs less than or equal to the new size can then be executed in the partition, and if the lower boundary has been raised, only self-relocating programs can be executed in the partition. If a foreground partition is lowered below 2K in size, the background partition is lowered below 14K in size, or the error occurred in the lower half of the background partition, the operator is informed that the partition is no longer usable.

A System Damage machine check interruption results in an attempt to record the error, followed by system termination (a check-stop state). The operator is informed of whether or not error recording was successful.

Channel Check Handler

CCH receives control after a channel error occurs and constructs an error record for RMSR to write in SYSRES. CCH passes the limited channel logout data and other pertinent status information to the appropriate I/O error recovery routine (ERP) unless analysis of the error indicates that system operation cannot continue (the error involved SYSRES, for example). If the ERP can correct the error, operations continue. If a permanent channel error exists, CCH records the error and cancels the partition affected. The operator is notified. System termination occurs if (1) a hard channel error occurs during the access of program phases or critical data contained on SYSRES, (2) two channels are damaged at the same time, or (3) more than four channel errors are outstanding concurrently.

The recovery support provided by the MCAR and CCH routines represents an extension of the facilities provided by the optional MCRR routine of DOS, which is available for System/360 models and which does not contain any repair or channel retry procedures.

ERROR RECOVERY PROCEDURES (ERP'S) FOR DOS VERSION 4

These device-dependent error routines are a standard part of the control program generated for any DOS environment. The limited channel logout provided by the DOS CCH routine is handled by a set of completely new CCH ERP routines. The DOS CCH ERP's are an addition to the current set of DOS ERP's. The latter are used without modification.

When a channel or an I/O device error occurs on a Model 145, the appropriate ERP is scheduled to perform recovery procedures. If the error is corrected, operations continue normally. If the error cannot be corrected (it is permanent), error recording occurs. If a user-written permanent error handling routine is not present, the affected task is abnormally terminated. The operator is notified of permanent I/O errors.

RECOVERY MANAGEMENT SUPPORT RECORDER (RMSR) FOR DOS VERSION 4

The recovery management support recorder (RMSR), new in DOS Version 4, is a generalized error recording routine that replaces the error recording functions provided in DOS Version 3. RMSR provides more comprehensive error recording than the recording routines in DOS Version 3 and writes records in SYSREC that are compatible in format with those written in the OS error recording data set SYS1.LOGREC. This enables a SYSREC file created by DOS Version 4 and a SYS1.LOGREC file created by OS MFT or MVT to be processed by the same Model 145 diagnostic routines. This capability was not previously available since a SYSREC file created by DOS Version 3 recording routines is not compatible in format with SYS1.LOGREC.

RMSR is given control to record the following types of error records:

- Records created by MCAR and CCH for permanent and intermittent machine and channel errors
- Unit check records that RMSR creates to provide statistics about permanent I/O device errors. This type of data is recorded by the OBR routine in DOS Version 3.
- Counter overflow records that contain statistics about the number of retries performed in correcting temporary I/O device errors. SDR handles this recording in DOS Version 3.
- Tape error statistics records that the TEBV routine records in DOS Version 3. The functions provided by the TEBV and EVA routines are handled by the tape error statistics portion of RMSR.
- IPL records that indicate the reason for this IPL and EOD records that are written when the operator issues the ROD command
- Miscellaneous data recordings for errors specific to a device, such as 3211 printer buffer errors and 3330-series disk storage buffer errors

ENVIRONMENT RECORDING, EDIT, AND PRINT PROGRAM (EREP) FOR DOS VERSIONS 3 AND 4

The EREP routine of DOS Version 3 is a special purpose utility that can be initiated as a job step via job control statements in the input stream or by an operator command entered via the console. Its function is to edit and print all error records contained in the SYSREC recorder file. EREP handles all status records written by DOS Version 3 Model 145 recovery routines and is included in all DOS Version 3 operating systems generated for the Model 145. Three versions are provided. One to print only System/370 records, one to print only System/360 records, and one to print both System/370 and System/360 records.

The EREP program supplied with DOS Version 4 is a totally rewritten version of the EREP program supplied with DOS Version 3. This EREP supports new functions and handles editing and printing of error records contained in a SYSREC file that was created by the RMSR routine, which is also a new feature of DOS Version 4. The new EREP does not handle error records written by DOS Version 3 MCAR, CCH, OBR, SDR, or MCRR routines.

The DOS Version 4 EREP program is self-relocating and System/370 model independent. It runs in the DOS Version 4 minimum batched partition size of 14K. Additional storage provided in the partition is used to improve the performance of EREP. The following facilities are supported:

- Editing and printing of the entire contents of the SYSREC file
- Selective editing and printing of the SYSREC file or SYSREC history tape contents (all MCAR records by CPU model, all CCH records by CPU model, all unit check records for each I/O device, etc.)
- Editing and printing of tape error statistics records from the SYSREC file or history tape
- Creation or updating of a tape-error-statistics-only history tape or a SYSREC history tape
- Summarization of tape error statistics data from the tape error statistics history file or the SYSREC file.
- Clearing of the SYSREC file
- Summarization of selected device errors from the SYSREC file

This new version of EREP can be initiated via the input stream or via an operator command as could previous versions.

CHECKPOINT/RESTART FACILITIES FOR DOS

Programs terminated because of an I/O device or channel error or as a result of a system termination can be restarted from a checkpoint or from the beginning of the job step if their job control is resubmitted with the appropriate restart control statements included. Malfunctioning I/O devices can be removed from the table of available devices by the operator, and different devices of the same type can be assigned to job steps via their job control or by the operator. Warm start facilities are not required, since DOS does not build work queues. (DOS POWER, which builds input and output queues, does provide a warm start facility.) DOS users should plan program and system restart procedures.

50:15 REPAIR FEATURES

The programmed repair features supplied are designed to minimize the impact of malfunction diagnosis and repair on system availability. Fault location and repair time should be reduced by:

1. Improved error recording. Both intermittent and solid hardware and I/O errors are logged at the time of failure if Model 145 RMS routines are present. More status information is available than is recorded for Models 30 and 40.

2. Online error diagnosis. Error diagnosis and repair can be performed concurrently with system operation in a multiprogramming environment

to avoid total system, direct access facility, or tape subsystem unavailability.

3. CE diagnosis routines. More exacting CE diagnosis routines are available.

   The following maintenance and diagnostic routines are provided:

   - Online Test Executive program (OLTEP) and Online Tests (OLT's) for operation under OS and DOS control to test malfunctioning I/O devices concurrently with system operation
   - Microdiagnostics for use in diagnosing malfunctioning field-replaceable units (stand-alone routines)

OLTEP AND OLT'S - OS AND DOS

OLTEP is designed to operate as a processing program under operating system control. It handles the required interface between the operating system and the device-dependent OLT's. One OLTEP is provided for operation under OS and another for execution under DOS. These two OLTEP's support functions not provided by the currently available OS OLTEP and DOS OLTEP programs for System/360.

The inclusion of OLTEP in an operating system is automatic for a DOS Version 3 or 4 control program generated for a Model 145 unless OLTEP is specifically excluded by a system generation parameter. OLTEP is a system generation option for OS MFT and MVT users. A standalone version of OLTEP, called OLTSEP, is available as well. OLT's are obtained from the software customer engineer.

OLTEP directs the selection, loading, and execution of device-dependent OLT's for the purpose of I/O device testing and error diagnosis. OLTEP is also designed to verify I/O device repairs and engineering changes.

As with any other job step, OS OLTEP is invoked with job control and executes with a user-assigned priority. A minimum program area of 28K is required for OLTEP operation in OS MFT and MVT environments. DOS Version 4 OLTEP, also invoked via job control statements, operates only in the background partition in a minimum of 14K and cannot be run with a 6K supervisor. The input stream or system console device can be used to supply the parameters required for test operations (devices to be tested, options desired, etc.). Both OS OLTEP and DOS OLTEP ensure the protection and security of user data files and storage in use while OLT's are operating. OS OLTEP also ensures that the devices to be tested are online or offline (as far as the operating system is concerned) as required by the particular device type.

The OLTEP's for OS and DOS have the new capability of being able to access history records describing previous I/O errors on the device being tested. In addition, multiple devices can be tested during one OLTEP execution. If a console is used to define the test run, the new prompting facility can be requested as an aid to the user supplying the definition.

OLTEP and the OLT's reside in a DOS core image library. In OS MFT and MVT environments, portions of OLTEP reside in both SYS1.LINKLIB and SYS1.SVCLIB, while the OLT's can be placed in a user-designated disk library (partitioned data set).

OLTEP and OLT's can operate concurrently with other executing jobs in a multiprogramming environment and provide online I/O device testing, eliminating the necessity for complete system unavailability while many types of I/O errors are being diagnosed.

MICRODIAGNOSTICS

The diagnostics provided are divided into several groups. They are the resident diagnostics, the Basic group, the Extended group, the Manual sections, and the Integrated File Adapter section.

The resident diagnostics, which are loaded as part of the basic system microcode, are the CPU Checkout and the Storage Scan/Storage Clear routines. The CPU Checkout routine is automatically executed following power-on reset, IMPL, IPL, and system reset operations to test certain CPU functions. The CPU Checkout routine stops upon detecting an error, and other diagnostic routines must then be executed to pinpoint the suspected malfunctioning replaceable unit. The Storage Scan and Storage Clear routines are executed by setting the diagnostic switch on the system control panel to the desired function.

The other microdiagnostics provided are fault-locating tests that attempt to identify the failing replaceable component. These microdiagnostics are distributed on console file disk cartridges. They are loaded from the console file and use the system console device for communication.

The Basic group is designed to be used before the Extended group. The initial sections of the Basic group are executed directly from the console file, rather than from control storage, and they require only a minimal amount of hard core circuitry to be operational. Using this basic circuitry, a "building block" technique of verifying the circuitry is employed, and eventually, control storage is loaded with the remaining Basic group diagnostics, which are then executed under normal CPU control. The Basic group detects and locates failures in the basic CPU and the system console device.

The Extended group tests other CPU hardware (retry, timers, machine check, etc.) and the channels. The tests are designed to be executed in a building block sequence also, so that untested hardware is never used in testing out another hardware section. The Manual sections require manual intervention and are designed to test system control panel functions, storage, and the system console device.

50:20 RAS SUMMARY

The degree to which an installation benefits from available RAS features depends in part upon their proper implementation. It is desirable for Model 145 users to design a system that includes RAS features and to become involved in the implementation and use of maintenance procedures and aids. Specifically, the user can:

- Include OLTEP and OLT's in his operating system (optional for OS users)

- Plan system and program recovery procedures (use of checkpoint/restart and warm start facilities)

- Have operating personnel perform normal hardware maintenance procedures, such as the periodic cleaning of tape unit heads. Proper system hygiene should be maintained, in general.

- Implement an effective program of operator training so that the number of system malfunctions that occur because of operator error is minimized

SECTION 60: PROGRAMMING SYSTEMS PREINSTALLATION PLANNING FOR THE MODEL 145 IN BC MODE

This section discusses transition from OS MFT, OS MVT, or DOS Version 3 on System/360 to OS MFT, OS MVT, or DOS Version 4 on the Model 145 in BC mode, respectively. Conversion to OS/VS1, OS/VS2, or DOS/VS is discussed in the optional programming systems supplement that describes the operating system.

60:05 GENERAL CONSIDERATIONS

The hardware, I/O devices, and programming systems support offered by the Model 145 make it an attractive system for both DOS Version 4 and OS MFT and MVT users. The Model 145 provides DOS and OS users with the potential for increasing the throughput of existing applications and the capability of expanding into new application areas on an improved price performance basis. Specifically, the Model 145 offers Model 30 and 40 DOS users:

- Increased internal performance, approximately three to five times the Model 40 and five to eleven times the Model 30, when operating in BC mode
- More byte multiplexer subchannels for expanding terminal-oriented applications and twice as many selector channels with a faster data rate
- The capability of attaching more and faster I/O devices, such as more 2314 and 3330-series disk storage, 320 KB tape units, and the 3211 Printer. The new features of the 3505 reader and the 3525 punch offer expanded capabilities in card processing.
- Integrated 1400/7010 emulation that provides advantages over and above those offered by CS/30 and CS/40
- Improved throughput potential and additional functions through use of more processor storage, increments of which can be added for less cost than on Models 30 and 40

In addition to the general uses stated in Section 1, additional processor storage can be used in a DOS Version 4 environment for the following:

- Installation of POWER II to provide unit record data transcription (card reader, printer, and punch operations) overlapped with production job step processing. DOS users with a version of POWER already installed can allocate more or larger I/O areas and handle more devices to improve performance. The remote job entry function of POWER II can be used also. A minimum partition size of 38K is required to support remote job entry in addition to data transcription.
- Execution of full function language translators such as ANS Full COBOL V3 (54K), the PL/I Optimizing Compiler that has OS PL/I F capability (44K), and FORTRAN IV (40K)
- Execution of the Tape and Disk Sort/Merge program, which offers significantly improved performance when larger amounts of processor storage are used

- Installation of time sharing, using the Interactive Terminal Facility (ITF)--a partition size of approximately 40K is required to handle 10 to 12 terminals

- Implementation of DATA/360-DOS for data entry and verification operations, using 2260 Display Stations instead of keypunches and verifiers

- Installation of GPSS (General Purpose Simulation System)

- Installation of IBM-supplied application-oriented programs (Type II and program product)

Model 145 DOS Version 4 users can also take advantage of new DOS facilities such as:

- Use of the time of day clock for more precise time of day values

- Multiple core image libraries for executable program (program phase) residence instead of a single library. Each batched partition in the system can have a private core image library assigned.

- The ability to compile source programs, link-edit object modules, and execute certain system service programs in batched foreground partitions. Previously, these functions could be performed only in the background partition. Most IBM-supplied language translators (Type I and program product) can be executed in a batched foreground partition, as can CSERV, MAINT, and DSERV service programs.

- Data file protection at open time for disk files to prevent unauthorized access to data by programs

- Problem determination aids (dumps, traces, etc.) that are designed to identify a system failure as one of three types--a system hardware error, a system programming error, or a user error--so that the failure can be pinpointed and corrected more quickly

DOS Version 3 users who find it desirable to convert to OS MFT or MVT with installation of a Model 145 will find the transition eased by use of DOS emulation under OS, a feature not available on System/360 models. For larger DOS users, the following are some of the attractive features offered by OS MFT and MVT that are not provided by DOS Version 3 or 4:

- Expanded multiprogramming--up to 15 processing programs (partitions or regions) operating concurrently with unit record data transcription (multiple reader interpreters and output writers)

- Priority and job class job scheduling for more efficient use of available system resources

- Dynamic resource allocation by the operating system of I/O devices and direct access space at program initiation time, and of processor storage and programs during program execution

- Expanded data management facilities, including device independence, a data set catalog, shared direct access device support, and graphic device support

- Program relocation at program load time by the control program

- Extensive job accounting data and resource usage statistics gathered by the control program (significantly more than are provided in DOS)

- Expanded teleprocessing support (TCAM) and time-sharing support (TSO and CALL-OS)

- Alternate and multiple console support, including graphic consoles

- Extended debugging capabilities as offered by the generalized trace facility (GTF)

In addition to supporting a wide variety of general and specific application programs, OS MFT and MVT also support Model 145 features such as extended precision, block multiplexing, rotational position sensing for the 3330-series and 2305, and a remote 3210 Model 2 as an alternate or an additional console. The Model 30 or 40 OS user can now use integrated instead of stand-alone 1400/7010 emulation and, like the DOS user, will benefit from the expanded hardware capabilities--faster internal performance, more and faster channels and I/O devices, larger processor storage, etc.--offered by the Model 145.

Because extensive hardware and programming systems compatibility exists between the System/370 Model 145 and System/360 models, most Model 30 and 40 users can upgrade to a Model 145 (in BC mode) with a minimum of effort. This is also true for users of ASP (Asymmetic Multiprocessing System) who wish to upgrade a Model 40 support processor to a Model 145. Essentially, no more effort may be involved in the Model 145 installation process for OS MFT (and MVT) users and DOS Version 3 users who do not convert to OS than is required currently to change from one operating system release to another, or to regenerate an operating system to include new hardware, new I/O devices, and more control program options. In most cases, the fact that an OS MFT or DOS Version 3 user is upgrading to a Model 145 should not add to the effort that would be required if new applications were to be added and system changes were to be implemented for a Model 30 or 40 upgrade to another System/360.

60:10 OS MFT AND MVT TRANSITION

A system generation must be performed using OS Release 20.1 or later in order to obtain an operating system designed to support new BC mode Model 145 features. The OS starter system can be used on a System/360 or a System/370 in BC mode, and a control program for either a System/370 (BC mode only) or a System/360 model can be generated. The existing system generation job stream can be used, with the following modifications, as appropriate:

- Direct access space allocation for operating system data sets will have to be adjusted as indicated in IBM System/360 Operating System, System Generation (GC28-6554). If a 3330-series drive or a 2305 is to be used as a system residence device, UNIT parameters in job control statements must be changed where necessary.

- Stage I input must be modified to reflect the Model 145 configuration, including the presence of any new I/O devices or features, such as the 3211, the 3330-series, the 2305, integrated emulator programs, etc. Other control program options, such as I/O RMS, OLTEP, and performance improvement features, can be included.

- FCB and UCB images should be added to SYS1.IMAGELIB if a 3211 Printer is included in the configuration. User-written output writer procedures should be modified to include these specifications.

An OS MFT or MVT operating system generated for the Model 145 includes the following:

- A nucleus designed to operate in the BC mode fixed processor storage area of the Model 145. MCH, CCH, OBR, and SDR routines are included.

- Control program support of block multiplexing and rotational position sensing, as discussed in Section 30, if requested

- Support of user-specified new I/O devices and Model 145 operator consoles, including a remote 3210 as an alternate or additional console

- Support of the interval timer and time of day clock

- Support of certain new System/370 instructions by Assembler F

- The required interface to the integrated emulator program specified at system generation, if any

If integrated emulator programs are to be used, the steps outlined in Section 40 must be taken in order to convert from a 1400/7010 system or from current stand-alone emulation procedures.

EXISTING OS PROCESSING PROGRAMS AND JOB CONTROL

IBM-supplied OS Type I processing programs and OS nonapplication-oriented program products (language translators, utilities, etc.) will run on the System/370 Model 145 (in BC mode) without alteration. Subject to the exceptions stated in Section 10:05, user-written OS processing programs and certain IBM-supplied industry-oriented (Type II and program product) programs for OS that operate on Models 30 and 40 will also execute correctly on the Model 145 (in BC mode). Modification and reassembly of existing user-written OS processing programs are not required unless new processing is to be added or existing processing is to be altered. Modification of the job control for these processing programs is required if I/O device type is changed (from 1403 to 3211, for example), if direct access space allocation changes, if a DCB parameter is to be altered, etc. Conversion from an MFT to an MVT environment will also necessitate job control statement alterations. I/O device type changes do not necessitate processing program alterations unless device-dependent macros have been used, data organization is changed, or a DCB parameter that is specified in the program is to be changed.

CONVERSION TO 3330-SERIES AND 2305 DISK STORAGE

Conversion from currently installed direct access devices to the 3330-series involves the same procedures that are required now to change from one disk device to another, say from 2311s to a 2314. Existing disk data sets can be placed on 3336 Disk Packs, using an IBM-supplied utility in most cases. Assuming that data organization is not changed, consideration should be given to altering the block size used and the amount of space allocated to the data set. The location and size of each type area in an ISAM data set should be altered, taking into account 3336 Model 1 Disk Pack characteristics. These changes can be made in job control statements.

Sequentially organized data sets (processed by QSAM or BSAM) and partitioned data sets can be copied from the source direct access device directly to the 3330-series drive, using the OS IEHMOVE utility. Or they can be copied to tape and then to the 3330-series drive, using the same utility (if the source direct access device type is not present in the Model 145 configuration).

Indexed sequential data sets can be copied directly from the source direct access device to the 3330-series drive, using the IEBISAM utility. Alternatively, they can be unloaded to tape and then reloaded, using the same utility. Changes to space allocation, etc., can be made via job control statements.

Direct organization (BDAM) data sets can be copied on a track-to-track basis from the source direct access device to the 3330-series drive, using IEHMOVE, or copied to tape and then to the 3330-series drive. If more records are to be placed on a 3330-series track than are on a source device track, the existing reorganization program can be used to transfer the data to the 3330-series drive, and the program may have to be changed. Reprogramming of the randomizing routine used in the reorganization, and in all processing programs that access the BDAM data set, is necessary if a relative track or actual address reference is used and fewer (or more) 3330-series tracks are allocated to the data set than before.

Subject to the restrictions stated in Section 10:05 and those indicated for BDAM data sets, existing executable processing programs can be used without change to process data sets that have been transferred to 3336 packs. Nothing need be done to job control for these programs if the cataloged procedures supplied with the language translators are used, as long as the 3330-series is specified as a SYSDA device at system generation. Otherwise, job control statements must be changed to request 3330-series devices and, optionally, any change to a data set characteristic, such as block size. RPS support, as described previously, is provided automatically.

User-written programs that use the EXCP level to access disk data sets that have been transferred to 3336 packs may have to be modified to reflect the characteristics of the data set on the 3336, a different number of records per track, a different number of tracks per cylinder, etc. All 2311 and 2314 CCW lists will operate on 3330-series drives except those that are device or channel time dependent and those that support the file scan feature, which is not available for the 3330-series. User-written 2311 or 2314 error routines will not execute correctly and must be modified. RPS commands have to be added by the user if this support is desired for programs that use EXCP. (Note that the XDAP macro includes support of RPS commands.)

Data sets currently located on 2303 Drum Storage can be placed on a 2305 facility, using a data set utility program. Unit specification in the job control statements of existing programs that will access the 2305 facility, instead of the 2303 drum, must be changed. Also, to reflect the larger capacity of a 2305 track, it may be necessary to alter the block size used. The latter also can be done via job control statement alterations (without program reassembly) unless block size was specified in the program itself.

CONVERSION TO THE 3410/3411 AND 3803/3420 MAGNETIC TAPE SUBSYSTEMS

As previously stated, existing tape processing programs and their tape volumes need not be modified in order to be used with 3410/3411 and 3803/3420 subsystems with equivalent features whenever the same recording modes are used. Existing 2400-series support is used to support the 3803/3420 subsystem. Therefore, existing job control statements for 2400-series tape unit jobs need not be modified to specify a 3420. However, job control statements that indicate a 2400-series tape unit must be modified to specify the 3410 when a 3410 replaces a 2400-series unit.

Whenever possible, seven- and nine-track NRZI mode tape volumes should be converted to 1600-BPI PE format to obtain the benefits of the higher density and the PE recording technique. In cases in which tape volumes must retain seven-track NRZI format, for interchange with other systems for example, use of a 3400-series tape unit offers improved tape reliability and subsystem serviceability as already discussed.

Conversion of seven- and nine-track NRZI tape volumes can be done gradually during production processing. That is, the old master input tape volume is read in on a 3400-series tape unit with the appropriate compatibility (Dual Density or Seven Track) feature while the new master output tape is written on a 3400-series tape unit in 1600-BPI PE format. Existing programs that process these converted tapes need not be modified unless an altered characteristic (recording mode or block size, for example) is specified in the program DCB. Existing job control for these programs must be altered to request a tape unit with the new recording characteristics and, if desired, to change existing DCB parameters such as block size, number of buffers, etc.

Tapes that cannot be converted on an as-used basis, such as program tapes or active reference tapes that are not rewritten when processed, can be converted by using a copy utility.

CONVERSION TO THE 3505 CARD READER AND 3525 CARD PUNCH

Existing user-written Assembler and high-level language executable programs for 2540, 2501, 2520, and 1442 readers and punches can be executed without change using a 3505 reader or a 3525 punch with comparable features. Programs that handle combined read/punch operations must be modified. This involves alteration of punch CCW lists if the EXCP macro is still to be used. If the newly defined access method support of combined operations is to be used, EXCP code must be removed, separate data sets must be defined, and the appropriate BSAM or QSAM macros must be added to replace the EXCP macro. Card data set DD statements that specifically request allocation of a reader other than the 3505 (UNIT=2540, for example) or a punch other than the 3525 must be modified.

60:15 PLANNING OPTIMAL SYSTEM PERFORMANCE, USING BLOCK MULTIPLEXER CHANNELS AND ROTATIONAL POSITION SENSING DEVICES

Block multiplexing, rotational position sensing, and multiple requesting provide the user with another tool that can improve total system throughput in the area of multiprogramming. However, the effectiveness of this tool for a given installation depends largely on proper planning for its use. This section indicates how to use block multiplexer channels and RPS devices more effectively.

The guidelines outlined indicate how best to configure a system with rotational position sensing devices, how to plan job scheduling, and what to consider when determining disk data set characteristics. Explanations follow the statement of each guideline.

All guidelines presented are not necessarily practical for all users. Each item should be evaluated in terms of the processing requirements and hardware configuration of an installation.

SYSTEM CONFIGURATION AND GENERATION

Guidelines for system configuration and generation are as follows:

1. Multiple 3330-series strings should be placed on a single block multiplexer channel.

   Performance improvement occurs (1) as a result of overlapping the rotational positioning time of disk devices and (2) because more

I/O requests can be initiated in a given period of time, since the channel is free more often. When many disk devices are active concurrently on a block multiplexer channel, there is more potential for such overlap.

2. Direct access devices with RPS should be placed on separate channels from I/O devices without RPS. Alternatives are as follows:

   a. If it is necessary to place non-RPS devices on the same block multiplexer channel with RPS devices, give first choice to non-RPS devices with a buffered control unit, such as the 3505 Card Reader, the 3525 Card Punch and the 3211 Printer. These devices disconnect from a block multiplexer channel during the relatively long mechanical portion of their cycle, thereby freeing the channel for other operations.

   b. Tape units should not be placed on a block multiplexer channel with RPS devices unless absolutely necessary, because channel disconnection does not occur during any of their channel operations. If this is not possible, try to plan job scheduling to avoid having jobs using tape units and jobs using RPS support active on a block multiplexer channel at the same time. If this is not feasible, try to assign very low-activity data sets to these tape units.

   A device without channel disconnect capability can monopolize the block multiplexer channel for relatively long periods of time, thereby preventing (1) the initiation of other I/O operations on the channel and (2) the reconnection and completion of disk RPS channel programs already in operation on the channel. For example, a direct access device without RPS retains use of the channel during its search operations as well as during its reads and writes. If the device is a 2314 and block size is half a track, the channel is busy for 25 ms on the average (12.5 ms average rotational delay plus 12.5 ms read/write) for each I/O operation started for the non-RPS 2314 facility. Even if the block size used is relatively small, the channel can still be monopolized by the non-RPS device if there is high activity on the device.

3. The 2305 facility normally should not be placed on a block multiplexer channel with any other device.

   Exclusive use of a channel ensures optimum performance of the 2305 facility as a system residence device.

The following should be noted as regards specification of priority and ordered-seek I/O request queuing options for RPS devices at OS system generation. The priority queuing option ensures priority I/O request initiation for the device, but because of first-come, first-served handling of I/O operations on the block multiplexer channel, this option does not ensure that priority device channel programs will complete sooner than other RPS channel programs that were started later on the channel. However, the objective of specifying the ordered-seek queuing option (minimization of arm movement on a disk drive) can still be achieved when using RPS.

JOB SCHEDULING

Guidelines for job scheduling on the System/370 Model 145 are:

1. If total system throughput improvement, rather than maximum individual job performance increase, is the objective, schedule

jobs that use RPS together such that the maximum number of RPS devices are active concurrently on each block multiplexer channel.

Greater overlap potential exists when more RPS devices are active concurrently on a block multiplexer channel. (See item 1 under "System Configuration and Generation".)

2. When jobs that use disk data management functions with RPS support are executed concurrently with jobs that do not use RPS support, the devices assigned to the former should be on different channels from devices assigned to the latter.

   Alternatively, if jobs using RPS and jobs not using RPS must access devices on the same block multiplexer channel concurrently, the jobs without RPS support should have high seek activity such that search and read/write time is small compared to seek time.

   Assume sequentially organized data sets and TCAM message queues are allocated to drives in the same 3330-series string on a block multiplexer channel. A QSAM job step and a TCAM job step are executing and access the data sets in the 3330-series string concurrently. Since RPS commands are not used for TCAM message queue processing, each 3330-series disk drive containing TCAM message queues acts like a non-RPS device and can monopolize channel time. Thus, the job steps that use QSAM can be delayed. (See item 2 under "System Configuration and Generation".)

3. Allocate a data set that will be accessed using QSAM or BSAM chained scheduling to a device on a channel without active RPS jobs.

   The chained scheduling technique is designed to keep a device active as long as record processing keeps up with record reading or writing. Thus, the channel can be kept busy for long durations, preventing the execution of any other I/O operation on the channel. Note that while QSAM and BSAM support concurrent use of RPS and chained scheduling for access to a disk data set, the performance attained by using chained scheduling alone will not be improved significantly by using RPS as well.

4. When data sets are being processed by an RPS access method in a multiprogramming or multitasking environment and disk device assignment is handled by the user rather than by the control program, allocate as many separate RPS devices as is practical.

   This approach allows the possibility of having more concurrent operations on these data sets and therefore more seek and rotational positioning overlap.

5. If a response-oriented RPS job operates on a block multiplexer channel concurrently with other RPS jobs, job scheduling should ensure that the number of jobs executing simultaneously is such that the performance desired for the response-oriented job can be attained.

   The performance of a block multiplexer channel is affected by the percentage of time the channel is busy searching and reading. The read or write of a particular record may be delayed one rotation because the channel is busy servicing another channel program. The probability of a particular record being delayed is a function of the percentage of channel busy time. As block multiplexer channel utilization increases, the probability that individual channel programs will be delayed increases. It is theoretically possible for the read or write of a particular record to be delayed indefinitely because the block multiplexer channel is busy searching for and reading other records. That is, utilization of more and more block multiplexer channel time will normally result in better overall performance but will increase the likelihood of delayed response from any one data set.

DATA MANAGEMENT PARAMETERS

Guidelines for the use of data management parameters are:

1. When organizing direct data sets to be processed using BDAM, use fixed-length standard records and a record reference that includes ID (relative block, relative track and ID, or actual address).

   RPS is supported only for fixed-length standard and VBS formats without key reference because record position must be known in order to calculate the sector number required for positioning. However, if a key reference or a variable record format is used, RPS support is provided for write verification and update (after retrieval) operations.

2. Use a large block size for sequentially processed data sets whenever possible, subject to the availability of processor storage for buffers.

   The use of RPS can provide performance gains for both short and long disk record blocks. However, use of large rather than short blocks reduces the total time required to read or write a given data set because less disk space is required and fewer I/O operations are necessary. Note also that total throughput for a given block multiplexer channel is improved by using blocks of equal (or nearly equal) size for all data sets being processed on the channel.

3. Use fixed standard records for QSAM and BSAM data sets where possible.

   The channel programs used for fixed standard records free the channel more often than when other record formats are used. A search for the previously read record is not used in order to locate the next sequential record when fixed standard records are read sequentially. The sector number of the next sequential record is obtained by including a READ SECTOR command at the end of the channel program used to read each record. Therefore, the SEARCH command specifies the ID of the desired record and the channel is free during the time it would otherwise have been busy searching for the previously read record.

   Channel time is reduced when fixed standard records are written because the operation required to calculate the remaining number of bytes on a track after each write is eliminated. (Note that the disk control unit is still busy erasing to end of track after formatting write operations even though the channel is freed after the data record has been written.)

4. Use multiple buffers with QSAM and BSAM.

   The availability of multiple buffers per data set lowers the probability that a task will have to wait for a particular record. QSAM is designed to initiate an I/O request whenever a buffer becomes available, thus keeping the channel queue as full as possible. When BSAM is used, the programmer must handle the initiation of I/O requests.

The following summarizes the advantages of rotational position sensing, multiple requesting, and block multiplexing.

- System throughput increases can be achieved when multiple sequential data sets are processed concurrently on a single block multiplexer channel (using QSAM, QISAM, or BSAM) because a higher effective channel data rate results.

- The number of block multiplexer channels required in a given system configuration can be fewer than the number of selector channels that would be required to handle the same amount of data, because more effective channel utilization is achieved by block multiplexing disk operations.

- The performance cost to an installation of verifying disk write operations is sharply reduced.

- The greatest throughput improvement results from use of rotational position sensing with high-activity, transaction-based processing, that is, with applications that include one or more large jobs that:

    1. Use direct processing (BDAM) with fixed-length standard or VBS records and a record reference that includes ID

    2. Require a multiple volume data base of small records

    3. Process many additions and updates and use write verify

60:20 OS MFT AND MVT PORTABILITY

To avoid multiple system generations, an OS MFT or MVT user with multiple Model 145 systems may wish to generate a single operating system that can be used on every Model 145 in the installation. This is possible under the same system hardware and I/O device configuration restraints that exist for System/360 models. During the IPL procedure, channels and I/O devices may have to be varied offline, partition sizes may have to be redefined, etc., when the operating system is used with a different configuration than was specified during system generation.

A user with both a System/370 Model 145 and a System/370 Model 135 or 155 or a System/360 model in an installation may also wish to generate one operating system that can be used on both models. This approach provides backup when one system is unavailable and can eliminate the necessity of multiple generations.

The system generation procedure is modified to allow generation of an operating system that is portable among System/370 Models 135 and up in BC mode. Specifically, the SECMODS system generation macro will be changed to cause the inclusion of all the required model-dependent routines (such as MCH, EREP, etc.) for both the primary CPU and each secondary CPU indicated by the user. When a System/370 model is the primary model, MCH can be specified in the SECMODS macro as the error recovery routine for each secondary model. At IPL, the appropriate model-dependent routines are initialized, based on the CPU identification.

Previously, only an SER routine could be specified for secondary CPU's and this support continues for System/360 models. Hence, an OS operating system can be generated to include multiple MCH routines for System/370 CPU's or multiple SER routines for System/360 CPU's; however, one control program cannot contain a mixture of MCH and SER routines (MCH for the Model 145 and SER for the Model 40, for example).

Portability of an OS MFT or MVT operating system between a Model 145 and a System/360 model, say 40 or 50, can be achieved by utilizing a multiple nucleus control program under the following general conditions:

1. The system hardware and I/O device configuration of both systems must be similar. For example, a Model 145 OS control program generated to support block multiplexing mode and RPS direct access devices cannot be executed on a system without such channels and devices.

2. The same control program, MFT or MVT, must be used for both systems.

3. Consideration should be given to the processor storage sizes of the two models when determining the size of the scheduler, language translators, and the linkage editor(s) included in the generated system.

4. Processing programs that are to run on both models must use instructions and features common to both systems. For example, an Assembler Language program that uses the new general purpose instructions for the Model 145 or byte orientation can be executed on a Model 155 but not on Models 40 and 50. In addition, the integrated 1400/7010 emulators generated for a Model 145 cannot be run on Models 155, 40, or 50.

In order to generate an MFT or MVT operating system that is portable between the Model 145 in BC mode and the Model 40 or 50, the following steps are required:

1. A complete system generation must be performed to generate an operating system for the Model 145. The IPL-time system/operator communication option must be requested so that options specified can be altered during IPL.

2. A nucleus generation should then be done for the alternate system. The model number specified (in the SUPRVSOR, SECMODS, CENPROCS macros, etc.) will be that of the alternate system, not the Model 145.

3. Additional link-edits must be performed to add model-dependent routines to the generated multiple-nucleus operating system. Specifically, SER and EREP model-dependent routines for the secondary system must be included, as appropriate.

4. If extended precision floating-point divide is used in processing programs, the following steps should be taken. SYS1.LINKLIB contains two divide simulation routines that are part of the extended precision floating-point simulator. One uses extended precision hardware, the other does not. When a full generation is performed for the Model 145, a calling mechanism is set to request the divide routine that uses extended precision instructions at execution time, since the Model 145 contains these instructions.

   Therefore, the divide simulation routine that does not use extended precision should be transferred from SYS1.LINKLIB to another library and given the same member name as the divide routine with extended precision instructions. When the operating system is executed on a Model 145, SYS1.LINKLIB should be used by extended precision programs. When the operating system is executed on a Model 40 or 50, the alternate library should be used.

Whenever a new program that is to execute on both systems is added to a library or if the Model 145 hardware configuration changes, the user must consider whether or not portability is affected. (Note that program products are portable from one system to another only under special licensing agreements.)

60:25 TRANSITION TO DOS VERSION 4

A system generation must be performed using DOS Release 25 Extended, 26, or 27 in order to obtain a supervisor that supports new Model 145 BC mode features. Restoration of the DOS Version 3 (Releases 25 Extended and 26) distribution tapes to disk, and generation of a DOS Version 3 supervisor can be done on a System/360 or a System/370 model to generate a Version 3 supervisor for a System/370 or a System/360 model. Restoration of the DOS Version 4 (Release 27) distribution tapes to disk, and generation of a DOS Version 4 supervisor can be done on a System/370 model only and not on any System/360 model. A DOS Version 4 supervisor supports System/370 models only and will not operate on a System/360.

The existing system generation job stream can be used, modified to reflect the Model 145 hardware configuration and the use of integrated emulator programs, as appropriate. Additional supervisor options can be selected as well.

A DOS Version 4 operating system generated for the Model 145 includes the following:

- MCAR and CCH routines to handle expanded machine check interruptions. RMSR is included to handle error recording and OLTEP is present also (unless explicitly excluded).

- Support of the new I/O devices (3211 Printer, 3330-series disk storage, 3803/3420 tape subsystem, 3411/3410 tape subsystem, 3505 card reader, 3525 card punch) and the new Model 145 console specified

- Support of the interval timer and/or the time of day clock, if requested

- Support of certain new System/370 instructions by Assembler D (14K variant)

- The required interface to integrated emulator programs

In general, the new supervisor will be larger than the one currently in use because of the automatic inclusion of MCAR, CCH, RMSR, and OLTEP routines and user selection of additional options. (The minimum DOS Version 4 supervisor size for the Model 145 is 14K.) This increase will be less for DOS supervisors that currently contain the optional MCRR, OBR, and SDR routines. A larger supervisor and the availability of additional processor storage in the Model 145 will cause partition starting addresses to change. Therefore, existing user-written, nonrelocatable DOS programs have to be link-edited relative to the new partition starting addresses and placed in a new core image library. If relocatable modules for these nonrelocatable programs are not available, reassembly of the source modules, as well as link editing of the resulting relocatable modules, is required. Existing user-written, self-relocatable DOS program phases can be copied directly from the old core image library to a new one.

Subject to the restrictions stated in Section 10:05 and at the end of this subsection, alteration of an existing source program is required only if new processing is added, if existing processing is changed, if there is a change in the I/O device types used in the program (ASSGN job control statements may also require changes), or if the program contains a user-written routine that depends upon a particular release of DOS for communication with the supervisor. (Conversion of emulated programs and files is discussed in Section 40.)

Existing printer programs that specify the 1403 Printer in the DTF (either by default or specifically) can be used without change for a 3211 Printer to support the same functions as before. Additional job steps that load the FCB and the UCB must be added to the input stream where appropriate. Source programs must be modified if support of any of the new 3211 features (retry after error, etc.) is desired or if a function or feature that was not used previously is to be supported on the 3211.

Conversion from 2311s (or 2321s) to a 2319 or 2314 facility involves transfer of the data to the new devices, and program and job control statement modifications. (The same procedures are required for conversion from 2314s to 3330-series drives.) The amount of program alteration required depends on the file organization used and changes made to take advantage of the larger capacity of the new disk pack (larger blocking factors, changes in ISAM area allocations, etc). In all cases, the DTF in the program must be modified to specify the new direct access device, and if block size is changed, I/O areas must be redefined. Device assignments and extents specified in the job control statements for these programs must be changed also. VOL, DLAB, and XTENT control statements are not supported for 3330-series files. DLBL and EXTENT statements must be used.

Sequentially organized files on 2311s or 2321s (processed by SAM) can be copied directly from the source direct accesss device to the new device, using the Disk-to-Disk or Data Cell-to-Disk utility. Alternatively, files can be dumped to tape and then loaded on the new disk pack, using disk-to-tape and tape-to-disk utilities if the source direct access device is not in the Model 145 configuration.

Indexed sequential and directly organized files (processed using ISAM and DAM, respectively) must be transferred to the new disk pack with a user-written program. The original creation or reorganization program with appropriate modifications can probably be used. The randomizing routine used in programs that process directly organized files must be modified if relative track or actual track address record reference is used and fewer or more tracks are allocated to the file than before.

User-written programs that use the EXCP macro to access files that have been transferred to new disk packs may have to be modified to reflect the characteristics of the file on the new pack, a different number of records per track, a different number of tracks per cylinder, etc. All 2311 CCW lists will operate on 2314/2319 disk drives except those that are device or channel time dependent. CCW lists for the 2321 will operate also, except when they contain a device-dependent command such as RESTORE. (All 2314 CCW lists will operate on 3330-series drives except those that are device or channel time dependent and those that support the file scan feature, which is not available on the 3330-series.)

The OS discussion of conversion of existing tape programs and their existing tape volumes for use with a 3410/3411 or 3803/3420 tape subsystem applies to DOS also, except that a change in a parameter, such as blocking factor, involves alteration of the program (tape DTF). In addition, recording mode changes involve job control (ASSGN) statement alteration only (not DTF modification).

Existing user-written Assembler and high-level language programs for the 2540, 2501, 2520, and 1442 can be executed without modification using a 3505 reader or 3525 punch with comparable features. Assembler and high-level language programs using a combined read/punch file must be modified to define two separate files. In all cases, job control ASSGN statements for both reader and punch programs must be altered to allocate a 3505 or a 3525, as appropriate.

BTMOD macros that include the optional OBR/SDR error logging facilities must be reassembled in order to operate with the RMSR routine provided in DOS Version 4. Further, RMSR and OBR/SDR in DOS Version 4 do not support error recording for terminals in QTAM programs. Therefore, the OBR/SDR parameter in the TERMTBL macro must be removed, and reassembly using DOS Version 4 is required.

Note that DOS Version 4 does not support System/360 models and therefore does not include the following support that is available in DOS Version 3: 10K variant of Assembler D, Compatibility Support/30 (CS/30), Compatibility Support/40 (CS/40), Autotest, and Vocabulary File Utility Program. In addition, job control program size and minimum batched partition size in DOS Version 4 are 14K (increased from 10K).

60:30 DOS VERSION 3 AND 4 PORTABILITY

A DOS Version 3 or 4 user with multiple Model 145 systems can generate a single operating system that can be used on each system, subject to restraints imposed by differences in hardware configurations and engineering change levels.

A single DOS Version 3 or 4 supervisor can also be portable among similarly configured System/370 Models 135, 145, and 155 because of the way System/370 DOS RMS routines are structured. (DOS support is not provided for the System/370 Model 165.) During IPL, the determination of the model in use is made by the initialization routine, using the STORE CPU ID instruction. Bits are then set in the supervisor that are tested during RMS routine execution to determine which routine should be used in those cases in which model-dependent routines exist.

To enable a supervisor for the Model 145 to handle the different length CPU extended logout areas for Models 145 and 155, a new parameter (PORT) has been added to the system generation CONFG macro. The PORT parameter affects only the logging of CPU extended logout data, not supervisor portability itself.

If a supervisor for the Model 145 is generated with a PORT macro that specifies the Model 155, when the Model 145 supervisor is used on a Model 155, the 672-byte CPU extended logout for the Model 155 will be recorded. If the PORT parameter is not included, CPU extended logout data for the Model 155 will not be recorded. When a Model 155 supervisor runs on a Model 145, the CPU extended logout area will be recorded. Since there is no CPU extended logout area on a Model 135, the PORT parameter is not required to achieve portability of a Model 145 supervisor to a Model 135. However, when a Model 135 supervisor runs on a Model 145, CPU extended logouts are disabled and recording of Model 145 CPU extended logouts does not occur.

A Version 4 supervisor for a Model 145 is not portable to a System/360 model; however, a Version 3 supervisor is. If a single Version 3 supervisor is to be used for both a System/370 Model 145 (in BC mode) and a System/360 model, say 30 or 40, Model 145 should be specified at system generation time so that MCAR and CCH are included. Model 145 MCAR and CCH routines will not execute properly on a Model 30 or 40. When a DOS Version 3 supervisor containing them is loaded during IPL, MCAR and CCH are disabled automatically by the initialization routine when the routine determines that it is not operating on a Model 145. (Note that MCAR and CCH cannot be disabled by the operator with the RF parameter when a DOS supervisor containing them operates on a System/370.) This means that the Model 30 or 40 will run as it would if the optional MCRR routine were not included in the supervisor. That is, there will not be any machine check error recording and the system will enter the wait state when a machine check interruption occurs. Thus, if the facilities provided by MCRR routines are desired for a Model 30 or 40 supervisor, a system generation must be done for each unique model in the installation so that the appropriate model-dependent machine check routine is included in each supervisor.

As covered in the OS portability discussion, processing programs that are to execute on the Model 145 in BC mode and a System/360 or another System/370 model must use instructions, features, and I/O devices common to both systems. In addition, integrated 1400 emulator programs are not always portable among System/370 models (see Section 40). (Note that program products are portable from one system to another only under special licensing agreements.)

60:35 USE OF OTHER PROGRAMMING SYSTEMS

Subject to the restrictions stated in Section 10:05, users of OS PCP, TOS, BOS, BPS, non-IBM-supplied control programs, or OS MFT and MVT and DOS control programs not generated for a Model 145 can execute their existing control and processing programs on a Model 145 (in BC mode) with a hardware and I/O device configuration comparable to that of the System/360 model now installed. However, since Model 145 RMS (machine check and channel check routines) are not included in these control programs, the Model 145 will operate under the following conditions:

1. Single-bit processor and control storage error correction is performed; however, a machine check interruption does not occur.

2. The IPL setting of the recovery mask bit disables the CPU for interruptions after microinstruction retry corrections.

3. Damage to the time of day clock, the clock comparator, the CPU timer, or the interval timer causes a machine check interruption condition and generation of logout data.

4. An unretryable or uncorrectable CPU error, a double- or multiple-bit processor or control storage error, or a storage protection failure causes an exigent machine check condition and generation of CPU extended logout data.

A machine check control switch, which determines the action that is to be taken when a machine check condition occurs, is present on the system console. When the switch is in the process position, machine checks cause an interruption and logout if they are enabled. This setting is to be used when an operating system containing Model 145 RMS is in operation. When the switch is in the stop after log position, all enabled machine check conditions cause an interruption and a logout, after which the system stops.

If the Model 145 is not set to stop after a machine check when an operating system without Model 145 RMS is used, machine check mask bit settings are established at IPL as described above and the system takes whatever action was planned for machine checks:

- Any control program without a recovery routine included (for example, SER0 or SER1 for OS or MCRR for DOS) will enter the wait state after a machine check interruption and logout. The logged data can be printed on the console (using display mode) or printed out with the stand-alone SEREP program. The operator can re-IPL and attempt to continue operations or the CE can perform diagnostic procedures.

- Any control program that contains a recovery routine will enter the routine and attempt execution. As stated in Section 10:05, these routines access model-dependent data and will not operate correctly. In addition, the logout data stored when the interruption occurred will have destroyed the code located in locations 128 to 511. Results are unpredictable.

Therefore, control programs containing non-Model 145 recovery routines should be run with the system set to stop after logging when machine checks occur.

For the following reasons, it is advantageous for Model 145 users to install an operating system that includes recovery management support designed specifically for the Model 145:

- The number of re-IPL's necessary because of machine malfunctions can be reduced. Many hardware errors can be corrected either by Model 145 hardware recovery procedures or RMS routines. The latter ensure the continuation of system operation whenever possible if the error cannot be corrected. This is particularly important during online operations. In Model 145 systems without RMS routines, recovery is provided only for correctable CPU errors and single-bit processor and control storage errors. Other errors will necessitate a re-IPL.

- Status information recorded by RMS routines will assist the customer engineer in the diagnosis of machine malfunctions. This data will be of greatest benefit in diagnosing intermittent errors.

- DOS users who install a DOS supervisor that includes Model 145 RMS will also have the advantages of integrated emulation. (CS/30 and CS/40 cannot be executed on a Model 145.)

- Versions of OS control programs that include Model 145 RMS are the only ones that include support of new Model 145 CPU features, block multiplexer channels, new direct access devices, and integrated emulators.

SECTION 70: COMPARISON TABLES OF HARDWARE FEATURES AND PROGRAMMING SUPPORT - SYSTEM/360 MODELS 30 AND 40 AND SYSTEM/370 MODEL 145

These tables have been included for quick reference. The first compares the hardware features of Models 30, 40, and 145. It also indicates DOS Version 4 and OS MFT and MVT support of Model 145 features. A dash (-) in a programming system column indicates that the hardware feature does not require programmed support. The second table compares DOS/VS, OS/VS1, and OS/VS2 support of Model 145 features and I/O devices.

70:05 HARDWARE FEATURES OF MODELS 30, 40, AND 145 AND DOS VERSION 4 AND OS (MFT AND MVT) SUPPORT OF THE MODEL 145

Hardware Feature	System/360 Model 30	System/360 Model 40	System/360 Model 145	DOS Version 4 Model 145	OS MFT and MVT Model 145
I. CPU	EC mode not implemented	EC mode not implemented	BC and EC modes standard	Supports BC mode only	Support BC mode only
A. Internal performance (BC mode)					
1. Relative to Model 40	-	1	3 to 5	-	-
2. Relative to Model 30	1	-	5 to 11	-	-
B. Instruction set					
1. Standard set (Binary arithmetic)	Standard	Standard	Standard	All languages	All languages
2. Decimal arithmetic	Optional	Optional	Standard	All languages except FORTRAN	All languages except FORTRAN
3. Floating-point arithmetic	Optional	Optional	Optional	All languages except RPG	All languages except RPG
4. Extended precision floating-point	Not available	Not available	Optional (Included in floating-point option)	Mnemonics in Assembler D (14K)	Assemblers F and H, FORTRAN H, FORTRAN H-Extended, PL/I Optimizing Compiler, and PL/I Checkout Compiler
5. New instructions (listed in Section 10:35)	Not available	Not available	Standard	Mnemonics supplied for user use in Assembler D (14K) for all except: LOAD REAL ADDRESS PURGE TLB RESET REFERENCE BIT SET CPU TIMER SET CLOCK COMPARATOR STORE CPU TIMER STORE CLOCK COMPARATOR STORE THEN AND SYSTEM MASK STORE THEN OR SYSTEM MASK. ANS Full COBOL Version 3 and ANS Subset COBOL have an option to generate CLCL, MVCL, ICM, and SRP instructions.	Mnemonics supplied for user use in Assemblers F and H for same instructions as DOS Assembler D. ANS Full COBOL Version 3 has an option to generate CLCL, MVCL, ICM, and SRP instructions.
C. Dynamic address translation	Not available	Not available	Standard	Not supported	Not supported
D. Channel indirect data addressing	Not available	Not available	Standard	Not supported	Not supported
E. Interval timer	Optional (16.6 ms resolution)	Standard (16.6 ms resolution)	Standard (3.33 ms resolution)	Supported for time of day (if time of day clock is not used) and for time intervals	Supported except for time of day requests

Hardware Feature	System/360 Model 30	System/360 Model 40	System/370 Model 145	DOS Version 4 Model 145	OS MFT and MVT Model 145
F. Time of day clock	Not available	Not available	Standard	Supported (as an option) for time of day requests	Supported for time of day requests
G. Clock comparator and CPU timer	Not available	Not available	Optional	Not supported	Not supported
H. Microinstruction retry by hardware	Limited	No	Standard	Both successful and unsuccessful hardware retries logged by RMSR	Both successful and unsuccessful hardware retries logged by MCH
I. Machine check interruption	Occurs on CPU, main storage, and certain channel errors. One mask bit controls this interruption.	Same as the Model 30	Occurs after corrected and uncorrected errors. There are five types of machine check and all are individually maskable.	Machine checks are logged. Recovery procedures are performed.	Same as DOS with some differences in recovery procedures performed
J. Fixed lower storage area size (including logout area for machine and channel errors)	139 bytes	324 bytes	704 bytes reducible to 512 if extended logout area is moved	Data logged is handled by RMS routines	Data logged is handled by RMS routines
K. Compatibility features (Optional unless otherwise indicated)	1.1401/1440/1460 2.1620 (mutually exclusive features)	1.1401/1460 2.1410/7010 3.1401/1440/1460 DOS Compatibility (for use with CS/40)	1.1401/1440/1460 2.1401/40/60, 1410/7010 3.OS/DOS Compatibility (standard)	1. 1401/1440/1460 integrated emulator program 2. 1401/1440/1460 and 1410/7010 integrated emulator programs 3. -	1. Same as DOS 2. Same as DOS 3. OS DOS Emulator program
L. CPU cycle time	750 nanoseconds 1-byte data flow	625 nanoseconds 2-byte parallel data flow	Variable from 202.5 to 315 nanoseconds 4-byte parallel data flow	-	-
M. Direct Control feature	Optional	Optional	Optional	Supported by MICR IOCS	Supported by the Real Time Monitor
N. Monitoring feature	Not available	Not available	Standard	Not supported except by an Assembler mnemonic	Supported by GTF and an Assembler mnemonic
O. Program event recording	Not available	Not available	Standard	Not supported	Not supported
P. Interruption for SSM instruction	Not available	Not available	Standard	Not supported	Not supported

Hardware Feature	System/360 Model 30	System/360 Model 40	System/370 Model 145	DOS Version 4 Model 145	OS MFT and MVT Model 145
II. STORAGE					
A. Processor (main) storage sizes	16K 24K 32K 64K	32K 64K 128K 192K 256K	- - - 160K 208K 256K 384K 512K	All are supported	All are supported (MVT requires 256K or more)
B. Processor storage cycle	1.5 microseconds for 1 byte	2.5 microseconds for 2 bytes	540 nanoseconds fetch for 4 data bytes 607.5 nanoseconds store for 4 data bytes 540 nanoseconds fetch for 8 instruction bytes	-	-
C. Processor storage validity checking	Parity checking on each byte - errors are not corrected by hardware.	Same as Model 40	ECC checking on a doubleword. Single-bit errors are corrected by hardware.	Corrected intermittent single-bit and uncorrected storage errors are logged by RMSR.	MCH logs errors as does DOS
D. Control storage	Card ROS	Tape ROS	Reloadable monolithic storage (32K-64K) with ECC	-	-
E. Byte-oriented operands	No	No	Standard	Programmers can use the byte alignment hardware facility in Assembler programs.	Same as DOS
F. Storage and fetch protection	Storage protect is optional. Fetch protect is not available.	Same as Model 30	Standard	Storage protect is supported.	Storage protect is supported
III. CHANNELS					
A. Byte multiplexer channel - up to 8 control units	Standard	Standard	Standard	Supported	Supported
1. Subchannels	96 for sizes 16K to 64K. A special feature permits systems with 32K or 64K to have 224 subchannels.	32K-32 64K-64 128K-128 192K-128 256K-128	Number of subchannels per system is not related to storage size. A total of 16,32,64, 128, or 256 subchannels is permitted with any storage size.	Supported	Supported

Hardware Feature	System/360 Model 30	System/360 Model 40	System/370 Model 145	DOS Version 4 Model 145	OS MFT and MVT Model 145
B. Integrated File Adapter	Not available	Not available	One can be attached to handle from 3 to 8 2314A-type drives (2319,2312,2313, 2318 units)	Supported (Same support as if a channel and a control unit are present)	Supported as for DOS
C. Selector channels	Optional 0-2	Optional 0-2	Channel 1 standard, 2-4 optional if no IFA. Channel 2 standard, 3 optional if IFA present.	Supported	Supported
1. Channel Word Buffer feature	Not applicable	Not applicable	Optional for all installed selector channels.	-	-
2. Maximum individual channel data rate	312 KB	312 KB	820 KB without word buffer 1.85 MB with word buffer	-	-
3. Block multiplexer mode	Not available	Not available	Optional, permits any or all installed selectors to operate as block multiplexers	Not supported	Supported
D. Channel retry data in a limited channel logout area after channel error, and I/O extended logout data	No	No	Yes	CCH routine passes limited channel logout data to ERP to perform a retry of failing I/O operation if possible. I/O extended logout data is recorded.	Same as DOS
E. Channel-to-Channel Adapter	Optional	Optional	Optional	-	-
IV. OPERATOR CONSOLE DEVICES	1. 1052 M7 Printer-Keyboard 15 cps (No alter/display mode)	1. Same as Model 30	1. 3210 Model 1 console with alter/display mode (15 cps)	Supported	Supported as the primary or an alternate console
		2. Additional consoles, such as display units, optional	2. 3215 Model 1 console with alter/display mode (85 cps)	Supported	Supported as above

Hardware Feature	System/360 Model 30	System/360 Model 40	System/370 Model 145	DOS Version 4 Model 145	OS MFT and MVT Model 145
			3. Optional 3210 Model 2 console remote with either (1) or (2) - no alter/display	Not supported	Supported as an alternate or an additional console
			4. Additional consoles, such as display units, are optional.	Not supported	Supported as additional consoles by MCS and DIDOCS options
V. I/O DEVICES					
A. 3211 Printer with tapeless carriage, UCS, and 18 additional print positions	Yes	Yes	Yes	Supported	Supported
B. 3803/3420 Magnetic Tape Subsystem	Yes except Model 7. Models 3 and 5 cannot be attached to a byte multiplexer channel.	Yes, except Model 7. Model 5 cannot be attached to a byte multiplexer channel.	Yes	Supported	Supported
C. 3410/3411 Magnetic Tape Subsystem	Yes	Yes	Yes	Supported	Supported
D. Other tape units	All except 2420 Model 7	All except 2420 Model 7	All can be attached.	Supported	Supported
E. Direct access devices (2311,2314,2303,2301, and 2321)	All except 2303 and 2301 drums. Only channel 1 can have 2314 facilities attached. (Includes 2314B1/2319B1 facilities)	All except 2301 drum. Either channel 1 or channel 2 (but not both) can have 2314 facilities attached. (Includes 2314B1/2319B1 facilities)	All except 2301 drum. 2314 facilities can be attached to channels 1, 2, and 3. (Includes 2314B1/2319B1 facilities)	2303 and 2301 drums are not supported.	All are supported except 2301 drum.
F. 3330-series with RPS and multiple requesting	No	No	Yes on block multiplexer channels (on selector channels also if multiple requesting and RPS are not used).	Supported as an I/O device without RPS and multiple requesting. Sixteen-drive addressing is supported.	Support includes RPS, multiple requesting, and sixteen-drive addressing.

Hardware Feature	System/360 Model 30	System/360 Model 40	System/370 Model 145	DOS Version 4 Model 145	OS MFT and MVT Model 145
G. 2305 facility Model 2 with RPS and multiple requesting	No	No	Yes on block multiplexer channels. The word buffer feature is required.	Not supported	Supported
H. 3505 Card Reader and 3525 Card Punch	No	No	Yes	Supported	Supported
I. Other devices: 1231,1285,1404, 1412,1418,1428, 1445,1827,2301, 2302,2319-A3,7340, 7772,1052-7,2150	Yes except 2301, 7340, 1052-7	Yes except 2301, 7340	No	-	-

70:10 DOS/VS, OS/VS1, AND OS/VS2 SUPPORT OF THE MODEL 145 OPERATING IN EC AND DAT MODES

Hardware Feature	DOS/VS	OS/VS1	OS/VS2
I. MODEL 145 OPERATING IN EC AND DAT MODES	One virtual storage up to 16 MB is supported. Up to five problem program partitions.	One virtual storage up to 16 MB is supported. Up to 15 problem program partitions. Up to 37 system task partitions.	One virtual storage of 16 MB is supported. Up to 63 problem program regions of which up to 42 can be TSO foreground regions.
A. Instruction set			
1. Standard set (Binary arithmetic)	All languages	All languages	Same as VS1
2. Decimal arithmetic	All languages except FORTRAN	All languages except FORTRAN	Same as VS1
3. Floating-point arithmetic	All languages except RPG	All languages except RPG	Same as VS1
4. Extended precision floating-point	Mnemonics in Assembler D (14K)	Assemblers F and H, FORTRAN H, FORTRAN H-Extended, PL/I Optimizing Compiler, and PL/I Checkout Compiler	Same as VS1
5. New System/370 instructions for the Model 145 (listed in Section 10:35)	All are supported by the DOS/VS Assembler. ANS Subset COBOL and ANS Full COBOL provide an option to generate four new general purpose instructions (CLCL, MVCL, ICM, SRP).	All are supported by the System Assembler. ANS Full COBOL provides an option to generate four new general purpose instructions (CLCL, MVCL, ICM, SRP).	Same as VS1
B. Interval timer	Supported for time of day (if time of day clock is not used) and for time intervals	Supported for timing facilities, except for time of day	Not supported
C. Time of day clock	Supported (as an option) for time of day values	Supported for time of day	Supported for time of day
D. Clock comparator and CPU timer	Not supported	Not supported	Supported for timing facilities except for time of day
E. Channel indirect data addressing	Supported (Note: Channel program translation is performed only for I/O device types that are supported by DOS/VS.)	Supported	Supported
F. Monitoring feature	Not supported except by an Assembler mnemonic	Supported by GTF and an Assembler mnemonic	Supported by GTF and an Assembler mnemonic
G. Program event recording	Supported by the SDAID/370 program for program debugging	Supported by the Dynamic Support System	Supported by the Dynamic Support System

Hardware Feature	DOS/VS	OS/VS1	OS/VS2
H. Interruption for SSM instruction	Not supported (CPU is disabled for this interruption	Supported	Supported
I. Compatibility features 1. 1401/1440/1460	1. 1401/1440/1460 integrated emulator program	1. Same as DOS	1. Same as VS1
2. 1401/40/60, 1410/7010	2. 1401/1440/1460 and 1410/7010 integrated emulator programs	2. Same as DOS	2. Same as VS1
3. OS/DOS Compatibility	3.	3. OS DOS Emulator program to emulate DOS Versions 3 and 4	3. Same as VS1
J. Machine check logouts	Supported by MCAR and RMSR	Supported by MCH	Supported by MCH
II. STORAGE			
A. Real storage sizes supported (112K, 160K, 208K, 256K, 384K, 512K)	All	All except 112K	384K - Minimum dedicated batch 512K - Concurrent batch and TSO regions
B. Byte-oriented operands	Programmers can use the byte alignment hardware facility in Assembler programs	'Same as DOS	Same as VS1
C. Storage and fetch protection	Storage protect only is supported	Storage protect only is supported	Storage and fetch protection are supported for all regions
III. CHANNELS			
A. IFA and selector channels	All are supported	All are supported	All are supported
B. Block multiplexer mode	Not supported	Supported	Supported
IV. OPERATOR CONSOLE DEVICES			
A. 3210 Model 1 console with alter/display mode (15 cps)	Supported as the primary console. (No alternate console support is provided in DOS.)	Supported as the primary console. (Alternate and additional consoles, including display devices, are supported as well.)	Same as VS1
B. 3215 Model 1 console with alter/display mode (85 cps)	Supported as the primary console	Same as (IV.A) above	Same as VS1

Hardware Feature	DOS/VS	OS/VS1	OS/VS2
V. I/O DEVICES			
A. 3211 Printer with tapeless carriage, UCS, and 18 additional print positions	Supported	Supported	Supported
B. 3505 Card Reader and 3525 Card Punch	Supported	Supported	Supported
C. 3803/3402 Magnetic Tape Subsystem	Supported	Supported	Supported
D. 3410/3411 Magnetic Tape Subsystem	Supported	Supported	Supported
E. Other tape units	Supported	Supported	Supported except for 2415 tape units
F. 2314/2319 facilities	Supported for system residence, data files, paging, and by POWER	Supported for system residence, data sets, paging devices, JES spooling devices, and SYSIN devices	Supported for system residence, data sets, paging devices, SYSIN and SYSOUT data sets, and SYSIN devices
G. 3330-series disk storage	Supported for system residence, data files, paging, and by POWER. RPS and multiple requesting are not supported. Sixteen-drive addressing is supported.	Same as (V.F) above. RPS, multiple requesting, and sixteen-drive addressing are supported.	Same as (V.F) above. RPS, multiple requesting, and sixteen-drive addressing are supported.
H. 2305 Model 2 with RPS and multiple requesting	Not supported	Supported for system residence, data sets, paging devices, and JES spooling devices. RPS and multiple requesting are supported.	Supported for system residence, data sets, paging devices, and SYSIN and SYSOUT data sets. RPS and multiple requesting are supported.

INDEX (Sections 01 to 70)

This page intentionally left blank.

SECTION 80: DOS/VIRTUAL STORAGE FEATURES

If required, the optional DOS/Virtual Storage Features Supplement is to be inserted here. Availability of this supplement will be announced.

This page intentionally left blank.

SECTION 90: OS/VIRTUAL STORAGE 1 FEATURES

If required, the optional OS/Virtual Storage 1 Features Supplement, GC20-1752, is to be inserted here.

This page intentionally left blank.

SECTION 100: OS/VIRTUAL STORAGE 2 FEATURES

If required, the optional OS/Virtual Storage 2 Features Supplement, GC20-1753, is to be inserted here.

This page intentionally left blank.

SECTION 110: VIRTUAL MACHINE FACILITY/370 FEATURES

If required, the optional Virtual Machine Facility/370 Features Supplement is to be inserted here. Availability of this supplement is to be announced.

This page intentionally left blank.

READER'S COMMENT FORM

A Guide to the IEM System/370
Model 145

GC20-1734-2

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 . . .

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

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
Department 812
1133 Westchester Avenue
White Plains, New York 10604

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)

GC20-1734-2

A Guide to the IBM System/370 Model 145 Printed in U.S.A. GC20-1734-2

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)