TECHNICAL SUMMARY

digital

DECsystem 10

DIGITAL EQUIPMENT CORPORATION
200 FOREST STREET
MARLBOROUGH, MASSACHUSETTS 01752

Printed in USA EE 21041 26/81 20

TECHNICAL SUMMARY

digital

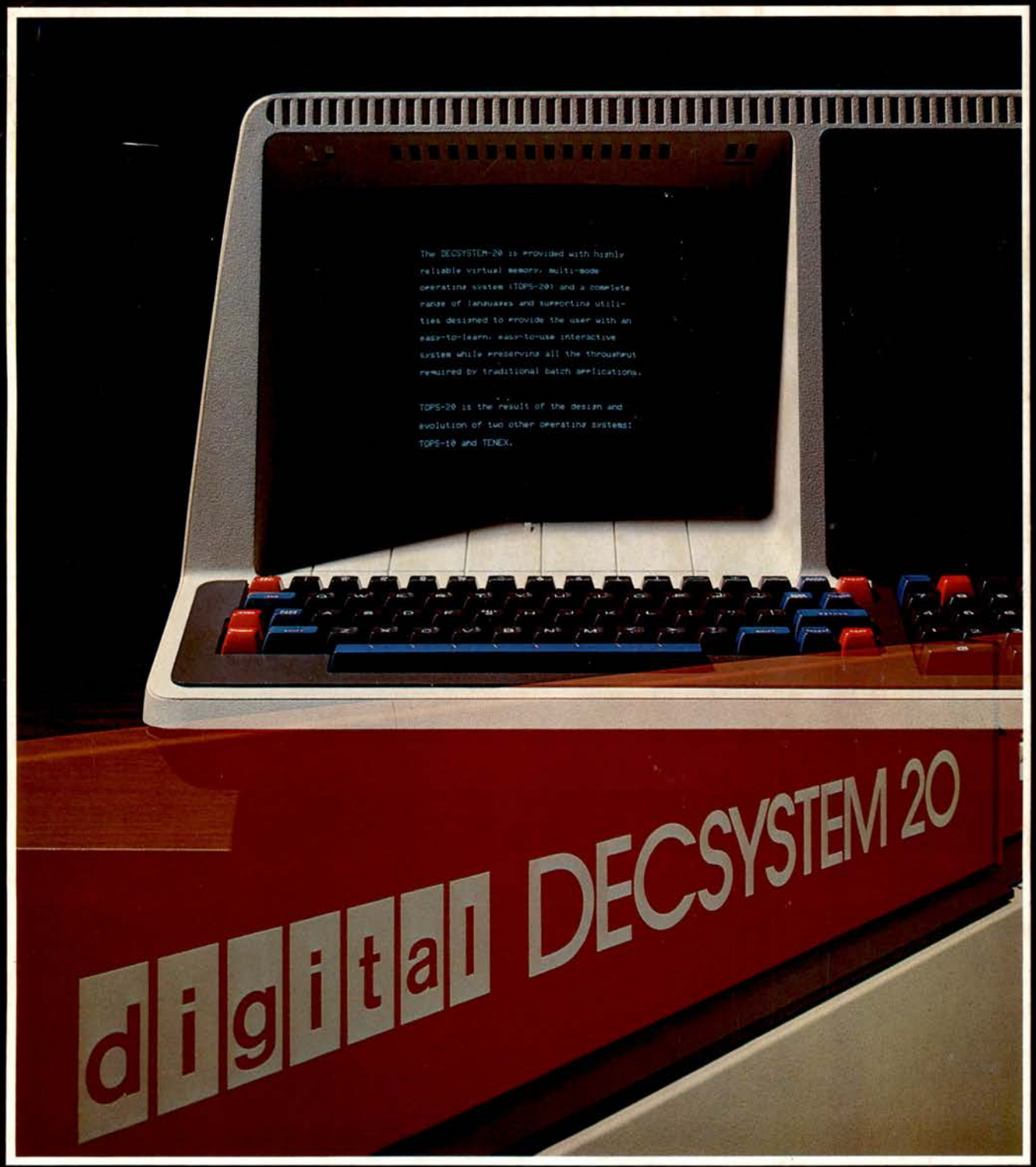
DECsystem 10

DIGITAL EQUIPMENT CORPORATION
200 FOREST STREET
MARLBOROUGH, MASSACHUSETTS 01752

Printed in USA EE 21041 26/81 20

digital

DECSYSTEM20 Technical Summary



The best of the big systems with the best of the small

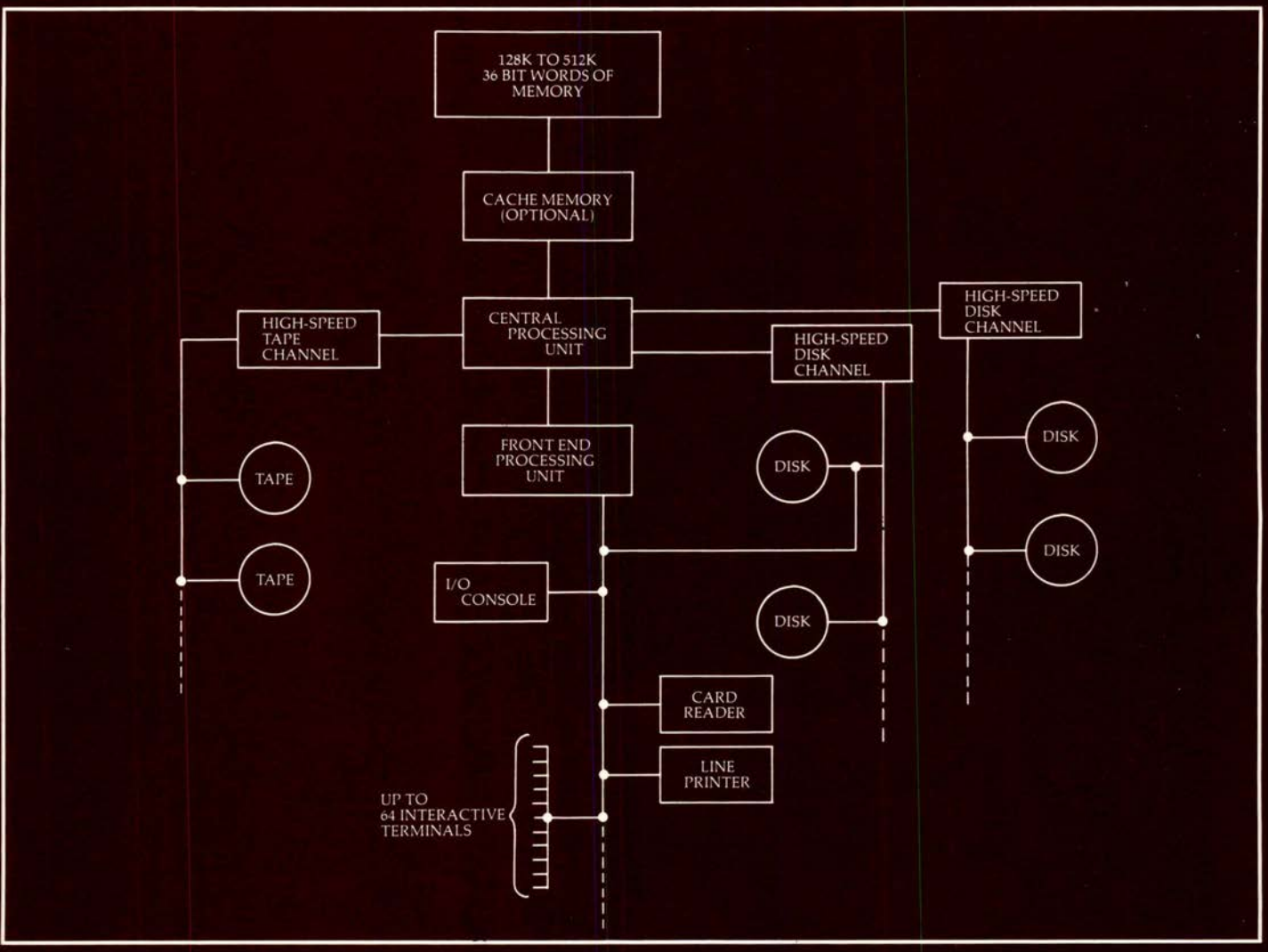


Table of Contents

DECSYSTEM 20 Introduction

Digital Equipment Corporation is pleased to present the DECSYSTEM-20—a medium-scale computer with large computer capabilities available at small computer prices.

The DECSYSTEM-20 features a high-performance, virtual memory system that provides a multi-tasking, multi-programming environment to support concurrent time-sharing, batch and transaction processing.

The DECSYSTEM-20 is easy to use and easy to maintain. It's flexible, expandable, reliable, efficient and above all—affordable.

The DECSYSTEM-20 is the result of over ten years experience in the design and development of large scale multi-language timesharing systems. It was in 1964 that DIGITAL installed the first.commercially available interactive timesharing system. And it was no accident.

Even then, DIGITAL wanted people to make better use of their time to be more productive, to get their "hands on" the computer.

Building on the success of these early systems, DIGITAL introduced the PDP-10 in 1967. It featured improved technology and better peripherals. The operating system had a better human interface so it was easier to use. Because of its success, development activity accelerated and in 1971, DIGITAL introduced the DECsystem-10 family of computing systems with a multi-mode operation system, called TOPS-10, offering time-sharing, batch processing, real-time and communications capability to support computer based remote stations.

Over 450 DECsystem-10 systems are now installed in 21 countries worldwide.

The design, manufacture and support of these large systems provided DIGITAL with the advanced technology experience necessary for the development of the DECSYSTEM-20.

DIGITAL also put its mini-computer expertise to work in the design of the DECSYSTEM-20. The central processor, memory, and I/O controllers are integrated into one neat, compact unit requiring a minimum amount of floor space—only twenty-eight square feet for processor, memory and I/O controllers. This cost-saving packaging is made possible because the DECSYSTEM-20 uses the latest in state-of-the-art technology. The processor is built of high-density, multi-layer circuit boards and emitter-coupled logic (ECL). Not only is the DECSYSTEM-20 smaller, but it's faster and more reliable.

The DECSYSTEM-20 features a large and powerful instruction set. The programs that the user writes will take advantage of this feature because the compilers and interpreters produce less machine code than do comparable systems. And because programs are smaller, they will run faster. The instruction set, despite its size, is easy to learn because it's logically grouped into "families" of instructions. One such family is the Business Instruction Set. It has sophisticated editing and character handling capabilities for program development or day-to-day problem solving. And the entire instruction set is micro-programmed because that's the cost-effective way to implement all these capabilities.

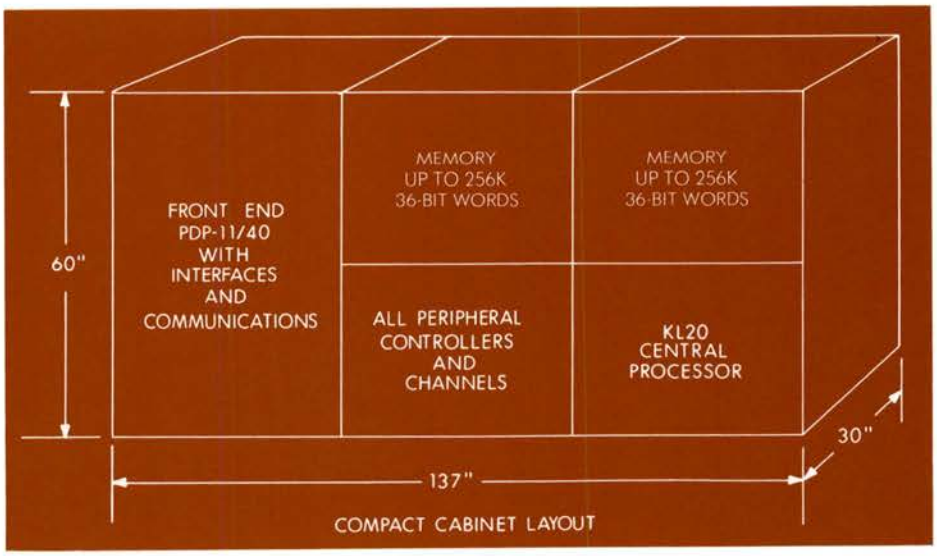
The DECSYSTEM-20 supports up to 2.4 million bytes of high-speed, low-cost memory. To ensure the reliability of operation and integrity of data, all memory read and write operations are parity checked.

The system features integrated high-speed data channels and mass storage controllers for disk and magnetic tape. These integrated I/O controllers permit the simultaneous transfer of data and control information. Each controller has its own direct path to memory; hence I/O operations on two separate controllers can be overlapped for greater throughput and efficiency. Since each controller may access up to sixteen words of data in a single request to or from memory, the DECSYSTEM-20 has an I/O bandwidth of more than 7 million bytes per second.

The central processor contains accounting and analysis hardware in the form of clocks and timers to provide accuracy and efficiency in the control and operation of the DECSYSTEM-20.

A PDP-11 based front end processor enables the central processor to do more work by handling unit-record peripherals, console operations, and terminal communications. Equally important is its role as diagnostic computer for the central processor. Even if the main system is completely inoperative, maintenance engineers can utilize the front end computer for module level diagnosis. Preventative maintenance may be done during timesharing operations. Both features increase system performance by reducing Mean Time to Repair and stand-alone preventative maintenance time.

The virtual memory hardware was designed in conjunction with the new TOPS-20 operating system to provide the most efficient and reliable system possible. As a consequence, programs that are larger than available core may be efficiently run without overlays. The paging hardware maps core for both the operating system and user programs, and provides information needed by the software to efficiently allocate core. Because the monitor is modular, demand paged, and write-protected, it is more reliable. The resident portion is quite small, thereby increasing system throughput by making more core available for user programs. User core is dynamically allocated as needed. No core is wasted since there are no partitions.



COMPACT CABINET LAYOUT

DECSYSTEM 20 Features Simplicity

Users may access the system by timesharing or through batch—whichever mode is the most appropriate to their needs. Production jobs may be entered into batch by cards or from a terminal. The same command and device specifications are used under both timesharing and batch. The user needs to learn one simple command language for both modes. Batch efficiency is high because the system can process multiple batch jobs at one time.

All user programs may communicate with each other in a variety of ways. Programs may be organized in a hierarchical control structure which provides greater reliability and security for user and system software. The failure of one program will not affect the others.

Data in the DECSYSTEM-20 file system is secure and reliable—by design. These features are especially significant because data and program sharing in memory is also a key feature. This controlled yet flexible data sharing facilitates powerful data base management systems.

Device independence and extremely flexible I/O access methods make programming easier and more efficient. Program execution is also enhanced since the I/O devices may be chosen at run time. Other features which increase system throughput include simultaneous update facilities and spooling of unit record devices such as the line printer and card reader.

Timesharing increases productivity by allowing the user to create programs using the on-line editor, and then immediately, from the same terminal, to compile and test them using on-line debugging techniques. Program development time is greatly reduced since many tests may be run in one session—no waiting for hours to see if there were errors. Data preparation is also much faster and easier since data files may be created on-line. Memos may be easily sent to all users via a mailbox facility, and documents may be formatted and printed using the document preparation program.

The command language provides easy access to these facilities. It prompts the novice user by guide words, explanations (if desired) and clear English error messages. At the same time, the expert user can enter commands quickly in an abbreviated form which will be recognized and acted upon by the command language processor. Thus, both novice and expert become more efficient and productive.

All DECSYSTEM-20 languages enhance program production. They contain source level debuggers, may be used under both batch and timesharing, and produce sharable, optimized object code. COBOL, FORTRAN, BASIC, APL, ALGOL, CPL, MACRO—the programmer may choose whichever is best suited to his application and expertise. ANSI-68 COBOL supports ISAM, high performance sorting, simultaneous updating, and DBMS—Digital's data base management system. COBOL utilizes the business instruction set for greater object code efficiency. FORTRAN, ALGOL and APL provide a variety of solutions to scientific problems, while BASIC, APL and CPL are designed and implemented to aid the beginning programmer.

The system administrator has full control over system access and usage and can determine who may use the system, what level of privileges they may have, and how the resources are allocated. Accounting tools enable him to accurately charge users for the resources they have actually used.

System operation is extremely simple. There is no monitor generation required. Operator procedures are very simple and the operator privileges are restricted to those necessary for normal system operation, enhancing system security and reliability. Flexible backup facilities enable single files to be saved and restored as well as the entire system.

The remote on-line diagnostic system enables maintenance engineers to do preventative maintenance during timesharing and to diagnose the system quickly and easily—thereby reducing Mean Time to Repair and enhancing system availability.

DECSYSTEM20 Hardware

System Architecture

The system architecture of the DECSYSTEM-20 is the result of design goals which included small size, high throughput, and high reliability and maintainability. System logic, housed in low profile cabinets, is made up of high density ECL circuitry on multi-layer boards. The Central Processor, memory, and mass storage controllers are integrated into a single cabinet assembly, requiring reduced floor space and cabling. Within the same assembly, a PDP-11 front end processor handles unit record devices, asynchronous communications, and system diagnostics. Mass storage devices are linked to the controllers over high-speed Massbus data and signal paths. Separate Massbusses provide increased throughput for tapes and disks. Unit record devices link directly to controllers attached to the PDP-11 UNIBUS. The overall design was heavily influenced by DIGITAL's advanced manufacturing technologies, developed and refined both in the mass production of minicomputers and in the production of the large-scale DECsystem-10. The resulting manufacturing techniques provide customers with a powerful hardware system and fully developed large computer software at low cost.

Central Processor Architecture

The Central Processor uses the word length and instruction set of the latest processor developed for the DECsystem-10 computer line.

The Instruction Execution Unit performs instruction execution, arithmetic operations, program-controlled I/O, and all clocking and metering functions performed by the DECSYSTEM-20. Included within the unit are eight sets of sixteen high-speed registers, a 1280 x 75 bit microstore for instruction logic, a 512 x 16 bit high-speed dispatch RAM, a seven level priority interrrupt system with vectoring, plus system clocks and performance meters.

The Memory Control Unit provides paths and logic for all transfers into and out of the DECSYSTEM-20 memory. Virtual-to-physical address translation is performed by a 64 x 154 bit RAM. The Memory Control also includes a series of multiple word channel buffers for high-speed direct I/O over the Channel Bus to the individual Massbus devices. For each device type, there is one channel containing a 15-word data buffer, a channel command word register, and a control word location pointer—effectively a program counter. The channels perform data transfer by executing channel programs which are set up in memory by device service routines.

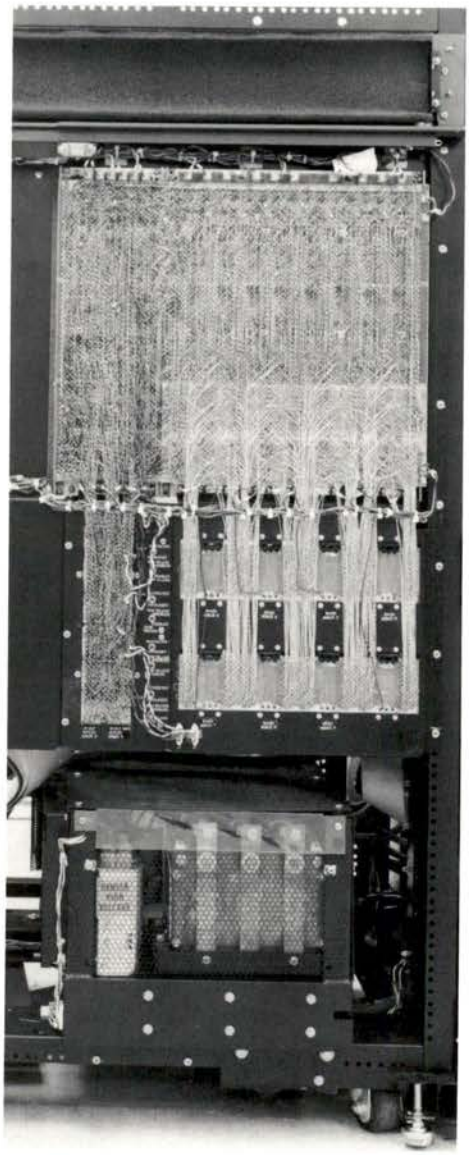
A Storage Bus provides a 1.6 million words/second transfer bandwidth between the Memory Control Unit and the DECSYSTEM-20 memory. In diagnostic mode, the Storage Bus is used to access the internal memory controllers. The Channel Bus is a physically short, high-speed data transfer path providing a peak I/O bandwidth of six million words/second between the Memory Control Unit and the Massbus controllers. The Channel Bus operates in a synchronous time division multiplexed mode, allowing the connection of multiple controllers to memory.

An Execution Bus connects the front end interface to the Instruction Execution Unit. The bus operates asynchronously with a bandwidth up to .5 million words/second for I/O instruction transfer.

A high-speed data path (Massbus) connects the integrated channel controllers and mass storage devices, such as disk or tape. Operating either asynchronously or synchronously, the Massbus transfers control and status information or blocks of data between devices and controllers. The UNIBUS, an integral part of the PDP-11 architecture, provides a data and status control path between the Front End Processor and the devices making up the PDP-11 system, including the communications interface and unit record I/O devices.

Dual access disks link directly into the PDP-11 disk controller for diagnostic and system initialization procedures. Integrated channel Massbus controllers allow simultaneous execution of non-data commands, such as seek, rewind, etc., while another device, linked to the same controller, is transferring data. To improve disk transfer rate, Massbus Controllers include a double buffer arrangement so that the software can pre-load the next transfer request during the current transfer. Using this capability, the system is able to read data corresponding to two separate requests from sequential sectors on the same disk without incurring rotational delay. Upon the completion of a channel operation, the controller generates a priority interrupt to the Central Processor. When the interrupt is recognized, control is passed directly to the appropriate routine through an interrupt vector supplied by the controller.

The Front End Interface provides for high-speed simultaneous two-way transfers of variable-byte data between the PDP-11 processor and the DECSYSTEM-20 Central Processor. The unit permits two-way doorbell-type interrupts between the processors and provides the access path to the Execution Bus for all diagnostic and bootstrapping functions.



Central Processor Features

The Central Processor directs the operation of the entire DECSYSTEM-20. It contains a microcoded instruction set, eight sets of sixteen general purpose registers, five accounting and performance meters, and instruction interrupt and trap facilities.

The Instruction Set is implemented in microcode, allowing for easy maintainability and expansion. The Instruction Set is a superset of that of the DECsystem-10 family of computers and includes 386 logically grouped instructions. A number of operation codes are also included with the Instruction Set which trap either to the TOPS-20 monitor or to the user's area, allowing a full range of programmable monitor calls. Although the number of instructions is large, they are easy to learn because of a logical grouping into families with consistent mnemonic code format. All instructions are capable of directly addressing a full 256K words of memory without resorting to base registers, displacement addressing, or indirect addressing. Instructions may, however, use indirect addressing and indexing to any level. Most instruction classes, including floating-point, allow immediate mode addressing, where the result of the effective address calculation is used directly as an operand in order to save storage and execution time.

Instruction Set

Half-word Data Transmission—These instructions move a half-word, optionally modifying the contents of the other half of the destination location. These 64 instructions differ in the direction they move half-words and in the manner they modify the other half of the destination location contents.

Full-word Data Transmission—The full-word data transmission instructions move one or more full words of data. The instructions may perform minor arithmetic operations, forming the negative or the magnitude of the word being processed.

Byte Manipulation—Four byte manipulation instructions pack or unpack bytes of arbitrary length anywhere within a word. Two instructions are provided for updating byte pointers.

Logic Instructions—The logic instructions provide the capabilities of shifting and rotating, as well as performing the complete set of 16 Boolean functions of two variables.

Floating-point Arithmetic—These instructions perform scaling, negating, addition, subtraction, multiplication, and division in single and double precision, floating-point format. In the single-precision floating-point format, one bit is reserved for the sign, eight bits are used for the exponent, and 27 bits are used for the fraction. In double-precision floating-point format, the fraction is extended to 62 bits in the second word. The sign bit of the second word is unused.

Fixed/Floating Conversions—Special instructions provide the capability for converting fixed-point numbers to or from floating-point numbers. Two sets of instructions are provided to perform this function: one set meeting the FORTRAN standard, a second set meeting the ALGOL standard.

Arithmetic Testing—The arithmetic testing instructions jump or skip depending on the result of an arithmetic test, and optionally perform an arithmetic operation on the test word.

Logical Testing, Modification, and Skip—These instructions modify, test, and/or skip using a mask of selected bits in an accumulator.

Program Control—This class includes conditional and unconditional jump instructions. In addition, it contains powerful instructions for efficient calls for short subroutines, calls for subroutines with many parameters and stack-oriented subroutine calls.

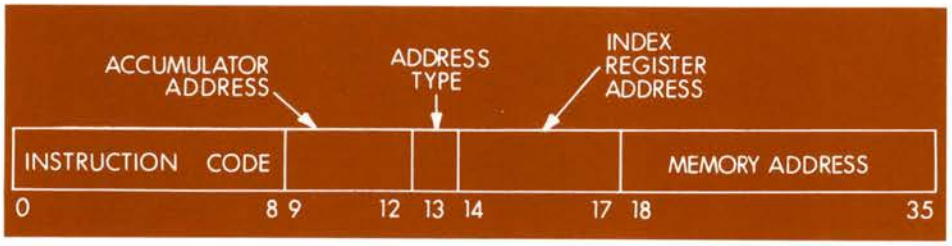
Input/Output Operations—Input/Output instructions govern direct transfers of control information to and from the peripheral equipment controllers and also perform certain operations within the processor. Block transfer instructions can handle multiple word data transfers to and from direct I/O devices.

Business Instruction Set—Five instructions comprise the Business Instruction Set in the central processor. Four are arithmetic instructions to add, subtract, multiply, and divide using double-precision, fixed-point operands. The fifth is a STRING instruction capable of performing nine separate functions, including:

- An EDIT Capability
- Decimal-to-binary and binary-to-decimal conversion in both offset and translated mode
- Move STRING in both offset and translated mode
- Compare STRING in both offset and translated mode (Offset Mode is byte modification by addition of the effective address of the STRING instruction to the source byte. Translated Mode is byte modification by translation through a table of half words located at the effective address of the STRING instruction. This also occurs in EDIT. In addition to providing the translation function, those instructions which use translation can control the flags in AC's and can detect special characters in the source.)

The Extended Business Instruction Set provides for faster processing of COBOL programs since there are specific instructions for doing more comprehensive string operations which commonly arise in such applications.

Instruction Format



The nine high-order bits (0-8) specify the operation; bits 9-12 usually address an accumulator but are sometimes used for special control purposes, such as addressing flags. The rest of the instruction word supplies information for calculating the effective address, which is used for immediate mode data or is the actual address used to fetch the operand or alter program flow. Bit 13 specifies the type of addressing (direct or indirect); bits 14-17 specify an index register for use in address modification (zero indicates no indexing); and the remaining eighteen bits (18-35) contain the address. The codes not implemented as instructions trap and are interpreted by routines included for the purpose.

Number System— Arithmetic instructions in the DECSYSTEM-20 use two's complement, fixed-point conventions to do binary arithmetic. In a word used as a number, bit 0 (the leftmost bit) represents the sign, 0 for positive, 1 for negative. In a positive number, the remaining 35 bits represent the magnitude in ordinary binary notation. The negative of a number is obtained by taking its two's complement. Zero is represented only by a word containing all 0's.

Fixed-point Arithmetic. Two conventions define a number as an integer (binary point at the right) or as a proper fraction (binary point at the left). In these two cases, the range of numbers represented by a single word is $-(2)^{35}$ to $(2)^{35}-1$ or -1 to $1-(2)^{-35}$. Since multiplication and division make use of double-length numbers, there are special instructions for performing these operations to yield results that can be represented by a single word for single precision operations.

The format for double-length fixed-point numbers is an extension of the single-length format. The magnitude (or its two's complement) is the 70-bit string in bits 1-35 of the high- and low-order words. Bit 0 of the high-order word is the sign, and bit 0 of the low-order word is ignored. The range for double-length integer and proper fractions is thus $-(2^{70})$ to $2^{70}-1$ or -1 to $1-(2^{-70})$.

Floating-point Arithmetic. A double-precision word consists of the sign, an 8-bit exponent and a 62-bit fraction. This gives a precision in the fraction of 1 part in 4.6×10^{18} and an exponent of 10 to a power of from -38 to $+38$.

Processor Modes— Instructions are executed in one of two modes. Programs operate in either User Mode or Exec Mode. In Exec Mode operation, all implemented instructions are legal. The monitor operates in Exec Mode and is able to control all system resources and the state of the processor. In User Mode operation, certain instructions are illegal, causing monitor traps when executed. Functions such as User Mode I/O are done through calls to the monitor, issued by the user program.

Executive and User Modes are further divided into two submodes each.

Processor Modes

User Mode

Public Submode	Concealed Submode
• User Programs • 256K Word Address	• Proprietary Programs • Can READ, WRITE, EXECUTE or TRANSFER to any location labeled Public
• All instructions permitted unless they compromise integrity of system or other users • Can transfer to concealed submode only at entry points	

Executive Mode

(Monitor)

Supervisor Submode	Kernel Submode
• Performs general management of system • Performs those functions that affect only one user at a time • Executes in virtual address space labeled Public	• Performs I/O for system • Performs those functions that affect all users

Fast Register Blocks—General purpose registers are another DECSYSTEM-20 feature that help improve program execution. Eight sets of sixteen fast integrated circuit registers can be used as accumulators, index registers, and as the first sixteen locations in memory. Since the registers can be addressed as memory locations, they do not require special handling. How the registers are used depends on how they are addressed in the instruction word.

Program switching time between register sets is 500 nanoseconds.

Different register blocks can be used by the operating system and individual users. This eliminates the need for storing register contents when switching from User Mode to Exec Mode.

Accounting and Performance Meters—Five accounting and performance meters are built into the Central Processor. These include:

Interval Timer. A Programmable source of interrupts with a range from 10 microseconds to 40.96 milliseconds

Time Base. A one microsecond relative time-of-day clock which is used by the monitor for system accounting.

Performance Analysis Counter. Designed as a tool for testing and evaluating the system, this counter monitors either duration or rate of occurrence of various hardware conditions and events.

Two Accounting Meters. An Instruction Processor Meter which measures the amount of the instruction processor time used. The information is stored in the User Process Table. While the default count is user time only, the monitor may also include interrupt time and/or Exec Mode time in this accounting. The second is a Memory Reference Meter, which measures user program accesses to core memory. This information is also stored in the User Process Table.

Trap Handling—Trap facilities exist for handling arithmetic overflow and underflow, push-down list overflow, and page failures. This trap capability is separate from the program interrupt system and can cause interpretation of the trap to be performed in either the user's or the monitor's address space as appropriate for the instruction code trapped.

Priority Interrupt System— The Priority Interrupt System is one of the most flexible that is available. Devices are assigned under program control to any one of seven priority levels through the dynamic loading of a 3-bit register within the device controller. A program can change the priority level of any device or disconnect the device from the system and later reinstate it at any other level. In the same manner, a program can set, enable, or disable any combination of levels with a single instruction. In addition, the program can assign some or all devices to the same level and generate an interrupt on any channel.

Program-assignable priority interrupt levels provide much greater flexibility than permanently hard-wired systems. An interrupt can cause the processor and the interrupting device to initiate one of several possible actions. In response to an "interrupt grant" signal, the device may supply a 33-bit word which is decoded as 18 bits address, 12 bits data, 3 bits function. The processor then does one of the following:

- Executes the instruction found at the supplied 18-bit address (vectored interrupt)
- Transfers a word into or out of the addressed location
- Increments or decrements the word at the addressed location

Priority level 0 is of higher priority than the seven programmable levels and is reserved for the Front End Processor. This level is invisible to the CPU software and is provided as an access to main memory for the console.

Cache Memory

The DECSYSTEM-20 features a 160 nanosecond access time cache as a memory buffer. Data being written to or read from memory is typically found already in the cache 90 to 95 percent of the time, thus giving the central processor an effective access time of 221.4 nanoseconds. Another feature of the "state-of-art" design cache memory is that, unlike contemporary designs, it does not require write-through to memory. This eliminates, for example, the necessity of writing back into main memory each value of an index in a loop comprised of only a few instructions.

The cache is organized in groups of four word blocks which are written back to main memory only when it is necessary to make room for another four word parallel fetch. A cache sweep feature allows main memory to be updated with all or selected parts of the cache.

Memory System

The Memory Control Unit contains the logic necessary for memory mapping between logical and physical addresses.

The memory address mapping hardware is integrated with the software to provide memory management totally transparent to the user. Physical memory is divided into 512 word pages. All monitor and user addresses are translated by hardware from the program's (virtual) address space to physical memory address space.

The high-order nine bits of the 18-bit virtual address define the virtual page number and are used to index into a Page Table containing the mappings between virtual addresses and physical core for either the active user or the operating system. If the physical page number is found in the 512 entry Hardware Page Table, the low-order nine bits of the virtual address are appended to it to form the complete physical address. If the virtual address is not found in the Hardware Page Table a computation is performed to determine the page mapping. The hardware page table is then loaded with the correct mapping. Two hardware registers internal to the central processor contain the page mapping information for locating the monitor's and active user's page maps in core. These maps contain information pointing to the pages for the operating system or active user, which are in core or out on the swapping device. The storage address pointers are of three basic types:

Private
the page belongs to only one user

Shared
the page is shared by more than one user

Indirect
points to another page map page where the pointer may again be one of the tree types

In addition to the physical page numbers in the page map, control bits are present to serve various functions. For example, one bit is used to indicate that the page is read only; a second is used to indicate that currently no physical page corresponds to

this virtual address slot. Another, whose function is to reduce disk channel traffic, is called a "written" bit and indicates whether or not a page need be written out to the disk or discarded when its physical space is needed.

Page level sharing among users is supported by a hardware Shared Pages Table, which holds the physical address of the shared page. This table, which is addressed through a hardware register, enables more efficient memory management because routines need to maintain just one pointer to a page, no matter how many processes are sharing it. Similarly, there is an Indirect Page Table for indirect pages.

Information about how long a page has been in core and the number of processes sharing it is also stored by the hardware in a Core Status Table to aid the monitor in memory management.

Process Tables— There are two types of Process Tables in core which are used by the software for both system and user management. These are the Exec Process Table (EPT) and the User Process Table (UPT). Each is one page long. The EPT is provided for the monitor and UPT for each user process in the system.

User Process Table

- Arithmetic overflow vector address. Overflow affects just the user and not the system, so it is handled in the user address space.
- Core and instruction processor usage accounting clocks. This information is kept for each user to aid in precise user resource accounting.

Exec Process Table

- Channel recording area. Information about channel status is maintained here.
- Front End Processor Communications Area. Used in communications between the Central and Front End Processor.

The physical memory of the DECSYSTEM-20 consists of from 64K to 256K 37-bit words. Memory increments are in 32K blocks, with up to four controllers each controlling 32K or 64K words. Two- and four-way interleaving of memory is set up through software settable switches at system startup time. Because the memories are integral to the Central Processor, memory accessing is faster than with comparable external memory architecture.

Front End Processor

The PDP-11 Front End Processor is one of the more significant features of the DECSYSTEM-20. The integration of the PDP-11 into the DECSYSTEM-20 Central Processor allows each processor to maintain its own integrity with minimal overhead. Additionally, functions are shifted to the PDP-11 wherever possible to improve overall system performance and reliability, and to provide an expandable base for future hardware and software.

Front End functions include:

- Bootstrap the Central Processor
- Drive all unit record equipment
- Service local and remote communications hardware
- Drive operator and diagnostic consoles
- Respond to Central Processor malfunctions
- Run diagnostic programs

The number of diagnostics that run under timesharing has been extended. This translates into more system availability to the user since service personnel need not take the system when repairs are needed.

To increase efficiency of service and minimize the delay that could occur if service is not onsite, a remote diagnosing capability is built in. Each system is equipped with an additional asynchronous line interface through which a remote service center, with permission of the system manager, can access the system. The service center has all the capability and diagnostic tests available as would local personnel. The result: reduced repair times and more customer satisfaction.

Diagnostic Capability — As a diagnostic computer, the Front End Processor can examine the data paths and control logic of the Central Processor even if that unit is completely inoperative. A Diagnostic Bus linkage permits automated testing procedures and allows diagnostic tests to be run that would be impossible using conventional techniques. There is less chance for human error, because maintenance engineers do not have to rely upon a malfunctioning Central Processor to diagnose itself.

The Front End Processor can sense the internal status of the central processor and control the operation of the central processor. Now, normal maintenance no longer involves time-consuming and error-prone reading of indicators and flipping of switches. All the maintenance and diagnostic procedures that require this on older computers are now performed by diagnostic programs in the Front End Processor.

Power Failure Detection and Recovery — If system power fails, a detection circuit senses the condition and causes an interrupt. The interrupt can trigger operation of a program, which saves volatile registers and provides an orderly system shutdown, allowing for later system restart with a minimum amount of lost time. A program-selectable automatic restart capability is provided to allow resumption of operation when power returns. Alternatively, a manual restart may be used.

Over Temperature Protection — Thermal sensors strategically placed within the equipment detect high temperature conditions and cause power shutdown. This, in turn, initiates the power failure interrupt.

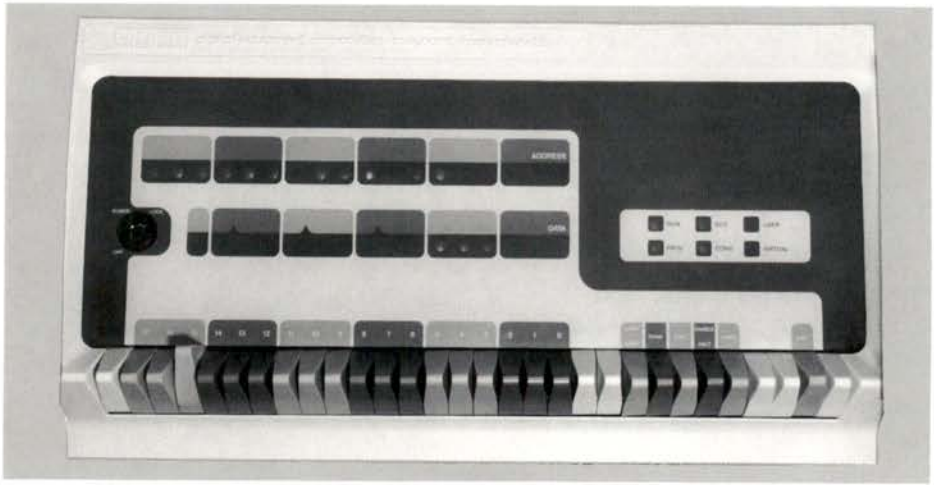
Self-Loading and System Startup — A cold start of the DECSYSTEM-20 requires a human being to push the load button and type in the time and date. It is always completed in less than five minutes. A warm restart after a temporary failure of either the DECSYSTEM-20 or the PDP-11 front end processor takes less than a minute and requires no human intervention.

DTE-20 — The DTE-20 console Front End Processor Interface connects the Central Processor and the Front End Processor to provide capabilities to interrupt, examine, deposit, diagnostic read and write, and transfer data during timesharing and diagnostic operations.

Console — The console terminal provides a direct system interface by simulating the control switch and display indicator functions for the Central Processor, thereby providing control of normal program and diagnostic operations and the ability to start, stop, load, modify, or continue a program. The console is connected for simultaneous two-way transmission through a full duplex asynchronous line interface to the Front End Processor.

Front End Processor — A PDP-11 is employed as a front end processor. It is equipped with a ROM for "cold starts", 28K words of core memory, dual floppy disks, console terminal interface, asynchronous line multiplexer, RP04 disk interface, and optionally, line printer and card reader interfaces.

The Front End Processor is equipped with a high-speed multiplexer for up to 64 asynchronous communication lines operating at speeds up to 9600 baud. The Basic System Package is equipped with a 16-channel multiplexer and 8 lines. Additional EIA compatible lines may be added in groups of eight. An optional 20mA line interface is available. The asynchronous line multiplexer meets EIA RS232C and CCITT (green book) V.24 standards.



Peripherals

A wide range of peripheral equipment is available with the DECSYSTEM-20. Each piece of peripheral equipment is fully integrated into the software system.

To increase throughput and storage capacity, a high performance disk system is provided. Removable storage elements insure maximum flexibility and convenience.

A reliable magnetic tape system operating in either of two recording modes provides data transportability and security.

To take full advantage of the interactive qualities of the DECSYSTEM-20, a high quality, high performance CRT terminal is offered.

Hard copy equipment includes DIGITAL's own widely accepted DECwriter II terminal, as well as high performance line printers in several models.

Two models of card readers are available which have been selected for their ability to work reliably with cards of varying condition.

Plotters and other applications related peripherals are available.

Today's computer systems place ever increasing emphasis on peripheral equipment and their development for the DECSYSTEM-20 is on-going.

Through this development, the user may be assured that the latest full capability peripheral equipment will be available.

RP04/RP06 Disk Systems

Up to four high-performance RP04 disk drives or two RP06 disk drives are supported. This provides an on-line storage capacity of 400 million characters. On the DECSYSTEM-20, the initial disk drive is dual-ported, linking to both the PDP-11 and the DECSYSTEM-20, channels. The diagnostic and initialization routines for the system are stored on this device along with the system software.

Each disk drive transfers data directly to and from memory under the control of a channel program. Because the controller provides overlapped seeking, the operating system will concurrently position two or more drives, shortening the effective access time and increasing throughput. This feature in combination with a rotational position sensing capability, can be used by the operating system to optimize disk throughput. A pre-load capability is provided in the controller so that when a channel program is completed the controller can send select and other control information to the disks without delay and initiate the new channel program.

The RP04/RP06 disk drive offers a high level of data reliability. A phase-locked loop clock system and state-of-the-art recording technique provide the utmost in reliable reading and recording. Additional error detection and correction hardware within the RP04 and RP06 disk drives provide information on the position and pattern of any error burst up to eleven bits within the data field. Correction of most data field errors is then accomplished under software control.

The ability to offset the read/write heads facilitates read recovery and increases data reliability and improves compatibility. Moreover, if a sector on a disk pack becomes defective so that attempts by hardware and software to recover data fail excessively, the operating system dynamically marks the sector as bad so that no future attempts will be made to use it. In most cases, the data can still be recovered so that the file can be copied to a new location.

Disk System Specifications

RP04

Disk Drive Capacity	20.48 million words
Peak Transfer Rate	5.6 microseconds/word
Access time:	
Track-to-track	7 milliseconds
Average	28 milliseconds
Maximum	50 milliseconds
Organization	128 words/sector
	20 sectors/track
	19 tracks/cylinder
	411 cylinders/pack
Number of Heads	20
Number of Recording Surfaces	19
Number of Disks/System	8
Number of Drives/Controller	4

RP06

Disk Drive Capacity	39.6 million words
Peak Transfer Rate	5.6 microseconds/word
Access time:	
Track-to-track	10 milliseconds
Average	30 milliseconds
Maximum	55 milliseconds
Organization	128 words/sector
	20 sectors/track
	19 tracks/cylinder
	815 cylinders/pack
Number of Heads	20
Number of Recording Surfaces	19
Number of Disks/System	8
Number of Drives/Controller	4

TU45 Tape System

Features

- 1600 bpi, 9-track
- Program-selectable recording at 1600 bpi phase-encoded or 800 bpi NRZI
- Data formats are industry-compatible
- Expandable to eight tape drives in a single system
- Vacuum column for tape buffer
- High reliability

TU45 Tape Drive

Up to eight dual-density tape drives may be provided. Drives are equipped with a dual gap head which enables writing and data checking on a single pass. An automatically activated erase head operates on writing.

A transport door protects the heads, capstan and magnetic tape and other tape path components from dust and other contaminants. A tape cleaner is provided and equipped with a contaminant removal system. Operator controls are accessible while the door is closed.

The drive is a single capstan unit with vacuum column tape storage system to buffer abrupt changes in velocity of the capstan from the supple and tape up motors.

The TU45 may operate in read mode in either forward or reverse direction. The tape system is controlled by an integrated channel and controller. The operating system sets up a channel program and initializes the channel-controller. Data transfers then proceed without further process or attention.

Specifications for Tape and Control

Main Specifications	
Storage medium	½" wide magnetic tape (industry std)
Capacity/tape reel	40 million characters (at 1600 bpi)
Data Transfer speed	120,000 characters/sec., max.
Drives/control	8, max.
Data Organization	
Number of tracks	9
Recording density	800 or 1600 bits/in., program-selectable
Interrecord gap	0.50 in., min.
Recording method	NRZI for 800 bpi phase-encoded for 1600 bpi
Tape Motion	
Read/Write speed	75 in./sec.
Rewind speed	250 in/sec.
Rewind time	115 sec. typical
Tape Characteristics	
Length	2400 feet, max.
Type	Mylar base, iron-oxide coated
Reel diameter	10½ in., max.
Handling	Direct-drive reel motors, servo-controlled single capstan, filtered positive pressurized tape compartment, vacuum type tape cleaner
Error correction	(Phase-encoding only), "On-the-fly" error correction for a single-track dropout
Magnetic head	Dual-gap, read-after-write
Drives/System	8

DC20 Asynchronous Communications Multiplexer

Features

- Speeds up to 9600 baud
- 64-character hardware buffer for received characters
- DMA Transmitter for each line
- Split speed transmitter and receiver speeds may be different
- Conforms to EIA RS232-C, CCITT specifications
- Full-duplex, half-duplex

The DC20 asynchronous multiplexer is an integral part of the DECSYSTEM-20 front end. The basic system contains a 16-channel multiplexer with 8 lines implemented. Expansion to 64 lines is supported.

The system uses 16 double buffered MOS/LSI receivers to assemble incoming characters. An automatic scanner takes each received character and line number and deposits the information in a 64 character first in, first out, buffer memory referred to as the Silo.

The Transmitter also uses one double buffered MOS/LSI for each line. These are loaded directly from the front end processor's memory by means of a single-cycle, direct memory transfer. The current address and data byte count are stored in semiconductor memory within the DC20.

EIA levels with modem control leads are provided. A modem control option is available.

LP20 Line Printers

The LP20A and LP20B series printers are 300 line-per-minute, 132-column devices with 64 or 96 character print sets. The printer contains a paper advance mechanism, top-of-form control, self-test capability.

A solid state vertical format control unit (VFU), static eliminator, and paper receptacle are provided. The printer is an impact-type using a revolving character drum and one hammer per two columns. Forms with up to six parts may be used for multiple copies. Included with the printer is a control unit interfaced to the Front End Processor. The control unit operates as an NPR (DMA) device on the PDP-11 UNIBUS.

Paper and inked ribbon pass between a row of hammers and a continuously-rotating metal drum. The drum surface contains 132 columns of all print characters. Data to be printed is received and stored in a full line buffer. Printing starts when a control character (line feed, carriage return, or form feed) is sent. If more than 132 characters are sent before the control character, an automatic line feed occurs and printing continues on the next line with no loss of characters.

Printing is accomplished by scanning the stored characters in synchronization with the rotating drum characters and actuating the appropriate hammer as the desired characters move into the printing position. A 132-column line is printed in two drum revolutions; the odd-numbered columns in one revolution and the even-numbered columns in the second revolution.

High performance printers, LP20F and LP20H, are offered for applications requiring increased printer throughput. Both 64 and 96 character versions are available with scientific or EDP character sets. The full 132-column width may be easily reduced by the user so that narrower paper widths can be used.

Operation is similar to the LP20A described above, but a line is printed every drum revolution. The vertical format control is by industry standard 12 channel paper tape.

Line Printer Specifications

	LP20A	LP20B	LP20F	LP20H
Controller	LP20	LP20	LP20	LP20
Columns	132	132	132	132
Printing Characters	64	96	64	96
Printing Speed	300 lpm	240 lpm	1250 lpm	925 lpm
VFU	Prog	Prog	Tape	Tape
UNIBUS Transfer Type	NPR	NPR	NPR	NPR

CD20 Card Readers

The CD20 card readers are rated and designed to meet varying throughput requirements and provide reliable, quiet, trouble-free operation. These low-cost card readers accept 80-column EIA/ANSI standard cards.

For fast throughput, the user can choose the console model CD20B, which processes 1,200 cards/minute or the table model CD20A, which processes 300 cards/minute and has a smaller hopper while still including the same basic features as the higher speed model.

CD20 card readers are designed to prevent card jams and keep card wear to a minimum. The readers have a high tolerance to cards that have been subjected to high humidity or to rough handling and are worn or otherwise damaged.

To keep cards from sticking together, the readers use a special "riffle air" feature. The last half-inch of cards in the input hopper are subjected to a stream of air which separates the cards and air cushions them from the deck and from each other. This action unsticks those cards attracted electrostatically and loosens those cards attached through torn webs or hole locking. It also separates cards that are swollen and stuck from excess humidity.

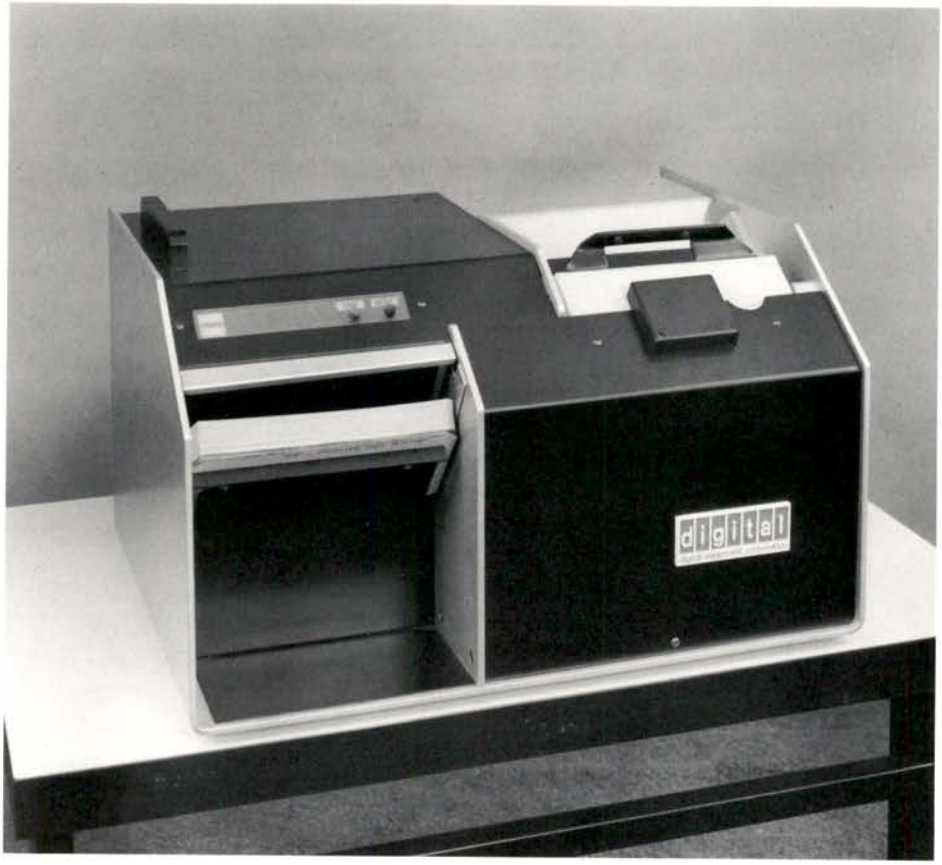
Cards entering the reader are selected through an advanced design vacuum picker. The picker and its associated throat block prevent the unit from double picking. To minimize the chances of jamming, the card track is short (less than four inches) so that only one card at a time is in motion.

The "riffle air" and vacuum picker features greatly extend card life. Stoppages are also reduced since the reader automatically tries six times before it determines that a card cannot be picked.

The read station of the CD20 card readers uses infrared light emitting diodes as its light source and phototransistors as its sensors to provide complete reliability. No adjustments are required during the ten-year life expectancy of the diodes.

Card Reader Specifications

	CD20A	CD20B
Hopper Cap	550	2200
Stacker Cap	550	2200
Speed	300 cpm	1200 cpm
Type	tabletop	Free-Standing



LA36
DECwriter II Terminal

Features

- Fast true 30-character-per-second output
- Low cost
- Up to six-part forms
- 132-column printing
- Variable-width forms
- Upper/lower case printing
- Field-proven print head
- ANSI keyboard
- Quiet operation
- Excellent character visibility
- Mechanical simplicity for high reliability
- Meets RS232C and CCITT (green book) V.24 Standards

The LA36 DECwriter II is a modular keyboard terminal designed for high performance, quiet operation, and high reliability at speeds of 10, 15 or 30 characters per second. The LA36 provides true 30-character/second printing for full utilization of a 300-baud communications line without the use of fill characters.

Printable characters are stored in a buffer during the carriage return operation; and while more than one character is in the buffer, the printer mechanism operates at an effective speed of 60 characters/second.

Adjustable pin-feed tractors allow for variable form width from 3 to 14-7/8 inches (up to 132 columns). The print mechanism of the LA36 will also accommodate multi-part forms (with or without carbons) of up to six parts with excellent print quality.

The LA36 keyboard generates the full 128 ASCII character set, consisting of 96 upper- and lower-case printing characters and figures and 32 control characters. In addition, the LA36 is available with optional ROMs (Read Only Memories) to accommodate special character sets, in addition to the standard ASCII such as the APL character set for use with the APL Programming Language or various foreign language character sets. Characters are printed using a 7 x 7 matrix with horizontal spacing of 10 characters per inch and vertical spacing of six lines per inch. The keyboard layout conforms to the ANSI standard and is similar to a standard office typewriter. The LA36 is particularly well suited to the office environment because of its exceptionally quiet operation. To insure clear visibility of the printed line, the LA36 incorporates a beveled cabinet top for an unobstructed view, positioning of the ribbon away from the paper surface, and a print head which automatically retracts out of the way when not in operation.

The LA36 was designed for reliability and ease of maintenance. It consists of a carriage system, a ribbon feed system, a paper feed system, and an electrical system. The three mechanical systems were designed for durability and high reliability with relatively few moving parts. The electrical system is contained on two printed circuit boards, employing the latest in integrated-circuit technology.

LA36 DECwriter II Specifications

Printing	True 30-character - per - second operation; 132-column format
Spacing	10 characters per inch horizontal, 6 characters per inch vertical
Paper	Up to 6 part forms, 0.020 inches maximum pack thickness. Variable width, 3 to 14⅞ inches Tractor Drive, pin feed
Keyboard	Standard typewriter which conforms to ANSI standard X4.14-1971 typewriter paired keyboard
Power Requirements	90-132 Vac or 180-264 Vac. 47-63Hz 300 W maximum (printing)
Interface	EIA RS232C, CCITT (green book) V.24 or 20mA current mode

VT52 DECscope Video Display Terminal

Features

- Upper and lower case
- ASCII
- 24-lines
- 80 characters per line
- ANSI/typewriter compatible keyboard
- Auxiliary keypad
- Direct Cursor Addressing
- Scrolling

The VT52, Digital Equipment Corporation's newest version of the DECscope, offers a combination of features not found in any other terminal. The VT52 is an upper-and-lower case ASCII video terminal whose display holds 24 lines of 80 characters.

The VT52 is upward-compatible with the VT50, but an identification feature allows software to distinguish between the two models. Software which uses Hold-Screen Mode to produce operator-controlled, screenful-by-screenful output to the VT50 will work perfectly on the VT52 without modification despite the different screen capacities.

When a key is depressed, a subtle click occurs, giving the operator an audible response for rhythmical keystrokes. For applications where absolute quiet is necessary, this sounder can easily be disabled. Anyone familiar with a standard typewriter can learn the basics of the VT52 operation in less than an hour because the VT52 keyboard is similar to a typewriter keyboard, rather than a teletypewriter; and DIGITAL provides thorough step-by-step documentation for both programmers and operators. Errors that might occur when the operator depresses several keys in rapid sequence are prevented by the VT52's three-key rollover feature.

The VT52 provides a "two-way" auxiliary keypad. In one mode, the keypad is used to generate program-compatible numeric codes. Applications which require much numeric input can use the VT52 without modifying hardware or software, while the operator uses the convenient "numeric pad".

VT52 Specifications

Display	Format: 24 lines x 80 characters Character Matrix: 7 x 7 Screen Size: 210mm x 105mm (8.3 in. x 4.1 in.) Character Set: 96-character displayable ASCII subset (upper and lower-case, numeric, and punctuation)
Keyboard	Character Set: Complete 7-bit ASCII set (128 codes) Key layout: Typewriter—rather than keypunch—format 63 keys Conforms with ANSI standard X4.14-1971 typewriter paired keyboard Auxiliary keypad: 19 keys; numerals, cursor-movement, 3 user-definable function keys
Audible Signals	Key-click: switch-controlled Bell: Sounds (a) upon receipt of control characters BEL: (b) when keyboard input approaches right margin (output from host approaching right margin does not cause bell to ring)
Page Overflow	LF causes upward scroll: Reverse Line Feed causes downward scroll
Cursor	Type: Blinking underline Control: Up or down one line; right or left one character; home; tab (fixed tab stops every 8 spaces); direct cursor addressing (allows cursor to be moved to any character position on the screen)

Functions	Erase display from cursor position to end of line; erase to end of screen; scroll up; scroll down
Hold-Screen Mode	Allows operator to halt transmission from host, preserving data on display. Operator can request new data, line- or screenful-at-a-time. Enabled/ disabled by Escape sequences sent by system software
Terminal Self-Identification	Terminal transmits on command a sequence unique to its model; software can identify features available on any terminal it is in contact with
Communications	EIA RS232C, CCITT (green book) V.24 normally supplied on VT52 ordered with DECsystem-20 20mA current loop available Transmission rates, full duplex (switch selectable) 75, 110, 150, 300, 600, 1200, 2400, 4800, and 9600 baud Switch-selectable local copy
Dimensions	Height: 360mm (14.1 in) Width: 530mm (20.9 in) Depth: 690mm (27.2 in) Minimum Table Depth: 450mm (17.7 in)
Weight	20 kg (44 lbs)
Operating Environment	DEC STD 102-Class B environment 10°C to 40°C (50°F to 104°F) Relative Humidity 10% to 90% Maximum wet bulb 28°C (82°F) Minimum dew point 2°C (36°F)

Or, software may place the VT52 in the alternate mode, in which each key on the keypad transmits a unique Escape Sequence. This allows the host computer to distinguish between keys typed on the auxiliary keypad and similar keys on the main keyboard. In this mode, each key on the keypad can be used to invoke a user-defined function.

The VT52 has a wide range of cursor-positioning functions. As well as moving the cursor one position in any direction, software can move the cursor to any position on the screen with a Direct Cursor Addressing command which specifies the destination for the cursor. The VT52 also offers fixed horizontal tabs, a "Cursor-to-Home" command, and two screen-erasure functions. Data on the screen scrolls up when a line feed function is performed with the cursor on the bottom line; it scrolls down when a reverse line feed function is performed with the cursor on the top line.

DECSYSTEM20 Software

The DECSYSTEM-20 is provided with highly reliable virtual memory, multi-mode operating system (TOPS-20) and a complete range of languages and supporting utilities designed to provide the user with an easy-to-learn, easy-to-use interactive system while preserving all the throughput required by traditional batch applications.

TOPS-20 simultaneously provides the features to support efficient full-language timesharing for program development and all types of interactive and terminal oriented applications, plus a full capability, multi-stream batch system.

All language processors are shareable and re-entrant to minimize memory requirements. Job control and command languages and all compilers are compatible under batch and timesharing.

The TOPS-20 operating system takes full advantage of the DECSYSTEM-20 paging hardware facilities to provide a virtual memory address space to all users of 256K words.

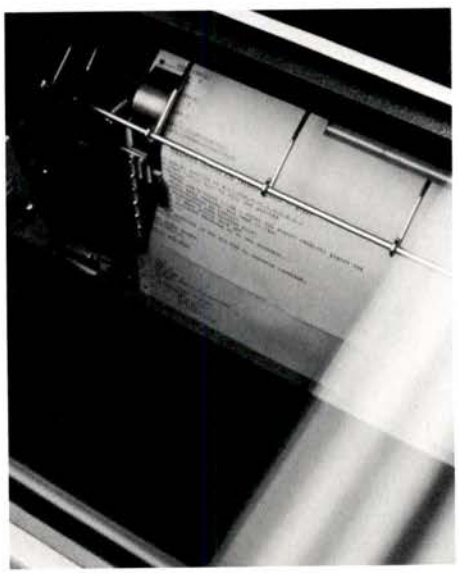
The system provides the concept of processes and jobs. Each job may create several processes with each process a separately schedulable entity with its own address space. Provisions are made for easy and efficient communications among processes at both the command and monitor call levels. The file system is highly developed and provides the extreme reliability so necessary in a multi-user system.

Facilities are provided for file sharing, file protection, and file backup. Files may be accessed by mapping into the user's address space. Conventional access methods are provided.

TOPS-20 is the result of the design and evolution of two other operating systems: TOPS-10, available on the large-scale DECSYSTEM-10 and proven in over 450 installations and TENEX, an advanced interactive and communications oriented system developed partly under US Government sponsorship for use with PDP-10 systems on the ARPA network.

TOPS-20 supports a full range of language processors, an easy-to-use powerful editor, a multi-stream batch system, an easy-to-learn, easy-to-use command language. It is a multi-purpose, multi-mode, virtual memory system with a highly reliable file structure.

Languages include FORTRAN, COBOL, ALGOL, BASIC, APL CPL and MACRO. Source level debuggers are provided for MACRO, COBOL and FORTRAN.



TOPS20 Operating System

Features

- Multi-User, Multi-Mode
- Multi-Language Interactive processing
- Multi-stream batch processing
- Communications
- Virtual Memory
- Multi-process job structure
- Easy-to-learn, easy-to-use command language
- Timesharing and batch command language compatibility
- Advanced reliable file system
- Front end processor operating system for communications and unit record control

In combining two existing components to form the basis for the DECSYSTEM-20, TOPS-20 provides all the features needed for maximum system reliability and ease of use. The operating system was designed to easily and efficiently serve many user tasks but be fully protected from errors they might cause. For example, the Command Processor runs in User Mode to isolate errors caused by users interacting with the operating system. The monitor itself is divided into resident and non-resident portions to efficiently use core and logically isolate functions for enhanced reliability. Special protective hardware processor modes are utilized in the implementation.

TOPS-20 supports a large number of interactive tasks together with concurrent batch processing. Extensive human engineering has gone into the system to make it easy to use and operate. A special prompting capability within the Command Processor assists the user

by completing on request the command being entered and by identifying argument and options available. Powerful commands allow routine tasks to be performed with a minimum of typing or training.

The TOPS-20 command language is used in both batch and timesharing modes. Any terminal user can construct a job under timesharing, and then submit the job to be processed by the batch system. Concurrent with the processing of that job, he may continue interactive work under timesharing. Reliability and ease of use were conditions imposed on all aspects of the DECSYSTEM-20 software development to arrive at a product which meets the needs of today's users.

The DECSYSTEM-20 operates in a virtual memory environment. TOPS-20 operation is process oriented. Each program invoked by a user is treated as a separate process, having its own 256K word addressing space. Processes are able to initiate other processes, suspend them, and communicate with them allowing for multiple concurrent tasks to be executed. This permits modularity in application development that can be exploited at execution time. For example, an application may have three distinct phases: (1) input data, (2) process data, (3) output results. By using a separate process for each phase, it is possible to be processing one set of data while another is being input and still another is being output. For the user who doesn't need such process structuring, its support by the system is transparent to him.

Timesharing

The DECSYSTEM-20 takes maximum advantage of system throughput capabilities, allowing many independent users to share system facilities simultaneously. The conversational, rapid-response nature of the DECSYSTEM-20 makes it particularly well suited for a wide range of tasks. The system supports a variety of CRT and hard copy terminals at speeds up to 9600 baud. Terminal users can be located at the computer site or at remote locations connected to the system through asynchronous communication lines. From a terminal, the user can control the running of a program; create, edit, and delete files; compile, execute, and debug programs. The user can also request assignment of peripheral devices such as magnetic tapes. When a request for assignment is received, the operating system verifies device availability and user-access privileges. The user is then granted private use of the device until he relinquishes it.

Timesharing on the DECSYSTEM-20 is designed in such a way that the command language, input/output processing, file processing, and process scheduling are independent of the programming language used. This allows use of a wide range of languages, such as COBOL, ALGOL, FORTRAN, BASIC, APL and MACRO.

GALAXY Batch Processing System

Features

- The same command language used in timesharing
- Multi-stream operation with operator control of the number of active streams
- Automatic line printer and card reader spooling with job accounting and control, including special forms scheduling
- Job dependency scheduling allowing creation of multi-job batch applications
- Job limits parameters to define a batch job's resource bounds
- Job Control Language error recovery facilities that allow command level recovery from expected or unexpected errors
- Support of all language processors, application tools, and system utilities
- User control of automatic job restart in the event of a system failure during job execution
- CPU time guarantees for groups of users

GALAXY enables the DECSYSTEM-20 to execute batch jobs concurrently with timesharing jobs. The batch user can enter a job into the batch subsystem using a card deck which contains control cards defining the command options for the job; or, using a terminal, he can create and submit a control file which is then intercepted by the batch subsystem and processed in exactly the same manner as the job submitted on cards. Because the control language for timesharing and batch are the same, a user is

able to duplicate any terminal session with a batch job by using the same set of commands and logic. Depending on the relative proportion of batch jobs on the system or their importance, the system administrator may assign a guaranteed percentage of CPU time for batch jobs.

The batch software programs include the Input Spoolers, the Batch Controllers, the Centralized Queue Managers, Task Schedulers, and the Output Spoolers.

The Input Spooler reads from the input device and requests the queue manager to enter jobs into the Batch Controller's input queue. The input data are separated according to the control commands in the input deck and placed into either user data files or the Batch Controller's control file, for subsequent processing. In addition, the Input Spooler creates the job's log file and enters a report of its processing of the job, along with a record of any operator intervention during processing. The log file is part of the standard output for the job.

The queue manager is responsible for scheduling jobs and maintaining both the Batch Controller's input queue and the output spooling queues. A job is scheduled to run acccording to external priorities, processing time limits and parameters specified by the user for his job, such as start and deadline time limits for program execution. The queue manager enters the job into the batch input queue based on priorities. After the job is completed, the queue manager schedules it for output by placing an entry in an output queue. When the output is finished, the job's entry in the output queue is deleted by the queue manager.

The output spooling program improves system throughput by allowing the output from a job to be written temporarily on the disk for later transfer to the printer. The log file and all job output are placed by the queue manager into one or more output queues to await spooling. When the printer is available, the output is then processed by the Line Printer Spooler. The Line Printer Spooler also includes such features as special forms control and character sets, multiple copies, and accounting information.

Operator Intervention—Normal operating functions performed by programs in the batch system require little or no operator intervention; however, the operator can exercise control if necessary. He can specify the system resources to be dedicated to batch processing, limit the number of programs as well as the core and processor time for individual programs, stop a job at any point, requeue it, and change its priorities. By examining the system queues, the operator can determine the status of all batch jobs. In addition, the batch system can communicate information to the operator and record a disk log of all messages printed on the operator's console. All operator intervention during the running of the Input Spooler, the Batch Controller, and the Output Spooler causes information to be written into the user's log file, as well as into the operator's log file, for later analysis.

Job Dependency—Although jobs are entered sequentially into the batch system, they are not necessarily run in the order in which they are read because of priorities set by the user or computed by the queue manager. Occasionally, the user may submit jobs that must execute in a particular order. To ensure that such jobs are executed correctly, the user can specify initial dependency counts in the control commands of the dependent jobs. Control commands in each job on which the dependent jobs depend must in turn decrement the count; and when the count for a dependent job becomes zero, it may begin execution.

Flexibility—The batch system allows the user great flexibility. The input spooler normally reads from the card reader, but can read from magnetic tape or disk. In the command string, the user can include switches to define the operation and set priorities and limits on core memory and processor time.

Error Recovery—The user can control handling of error conditions by including special commands in his job. These commands, copied into the control file by the Input Spooler, specify the action to be taken when a program encounters a fatal error condition. If an error occurs and the user has not provided for error recovery, the Batch Controller initiates a standard dump of the user's core area and terminates the job. This dump can be used to help debug a program.

Defaults—Although the batch system allows a large number of parameters to be specified, it is capable of operating with very few user-defined values. If a user wishes to specify the nor-

mal choice for the installation, he may do so by omitting the parameters altogether. These defaults can be modified by the individual installations.

Multi-stream Capabilities— The batch system can run multiple batch jobs concurrently—a feature which enhances both system and user throughput, since a user can separate jobs into several parallel job steps.

Command Processor

The TOPS-20 Command Processor serves as the user's interface to the system. Both batch and interactive users work with the same command processor because the command language is common to both. This consistency is important because it allows switching between the two modes easily. Equally significant is the time saved to learn only a single set of commands: new user training is much simplified.

Command consistency throughout a system is quite useful, but consistency of complicated, verbose, or hard-to-remember commands is still unacceptable for the convenience and efficiency required of a computer system. TOPS-20 was designed to provide the user with concise, meaningful commands that specify what he wants done while isolating him from details better left to the system.

As an example, consider a user who has a program sample to test. In the simplest case he types EXECUTE SAMPLE. With one short command he causes TOPS-20 to compile, load, and execute his program. He doesn't concern himself with details; he doesn't pre-allocate memory disk storage or even specify where the program is to be found. TOPS-20 and the command processor cooperate to make the user's job easy. The design philosophy is that a computer should be convenient for the user—not vice versa.

To continue the example, suppose the user also wants a compilation listing of his program. To specify the listing requirement, the user types
EXECUTE SAMPLE/LIST
which causes the same action as before but additionally produces the desired listing.

We can carry the example further. Suppose the user is familiar with the system and doesn't like to type EXECUTE every time he wants to carry out the compile/load/execute sequence. TOPS-20 requires that the user type only enough to uniquely identify any command, so it is sufficient to type only EX SAMPLE to be equivalent to EXECUTE SAMPLE. Suppose further that the user is testing SAMPLE and goes through the EXECUTE sequence frequently. He can type only EX any time after the first EXECUTE SAMPLE, because EX uniquely identifies EXECUTE—TOPS-20 remembers and assumes SAMPLE until explicitly given a different name.

To help the user who is not fluent in all details of TOPS-20, the question mark, when typed in a TOPS-20 command, will give a list of the alternatives available. For example, typing "CH?" will make TOPS-20 type a list of all commands beginning with "CH".

Abbreviated commands are useful, but the idea is extended in TOPS-20 by allowing the user to give an abbreviated portion of a command and press the ESCAPE key. If the command is unique, the rest of the command will be typed by TOPS-20 and will include guide words to prompt the user for the arguments to the command.

Suppose a user is beginning to learn the abbreviations for commands, but has not acquired full confidence with all the abbreviations; he can type "DIR", the abbreviation for the disk directory listing command. If he has doubts that this is what he wants, typing the control character ESCAPE will make TOPS-20 type out the rest of the command that it understands. In this case the extra characters typed will complete the command line to read "DIRECTORY (OF FILES)" to indicate the full command TOPS-20 understands. This gives the user the reassurance of seeing the full command without requiring him to type it out completely.

This feature, called command recognition, works on command arguments and file names, so that whenever a user has doubts about TOPS-20's understanding of his desires he can type the control character ESCAPE and get a full description of TOPS-20's understanding of what has been commanded without actually executing the command.

DECSYSTEM-20; however, the framework in which the command processor runs is equally useful. The TOPS-20 command processor is not a part of the operating system. It runs as a part of each user's job. This is possible because a job in TOPS-20 can consist of multiple tasks or processes (programs) associated with each other in a hierarchical structure. The command processor in TOPS-20 runs as the controlling process in a job.

The ramifications of this organization are many-fold. First, since the command processor is not a part of the operating system, the operating system is smaller. System operation is also more reliable because a command processor error affects only the job which exercised the bug; other jobs continue normally.

Second, the design of the system permits a job's command processor to control a task which is itself a command processor. Thus, new or special purpose command processors can be written, tested and used just like other programs, during normal system operations; stand-alone system time or operating system modifications are almost unnecessary.

Because of the program and data sharing supported on the DECSYSTEM-20, system efficiency is maintained by allowing multiple authorized users simultaneous access to a single copy of any program. Hence, the system and special command processor can be shared.

Implementation of TOPS-20 style commands and formats—recognition, abbreviation, prompting, help—is facilitated by table-oriented operating system calls (JSYS) which provide both consistency and reduced development time and effort.

In summary, TOPS-20 is engineered to be powerful yet easy to use and learn, and provides the structure and facilities required by computer users.

TOPS20 File System

Features

- Named files
- Controlled sharing among users
- Redundant root directories
- Write and close operations occur in correct order—files are always safe when reported safe
- Index Blocks kept in core for speed
- Integrated file and process space for convenience and security
- Disk and file space dynamically allocated, no preallocation required
- Complete dumping and back-up facility

TOPS-20 provides a system of named files with controlled sharing and protection among users. Data sharing is provided on a per page basis.

Reliability of the file system is ensured by design as well as by testing. File root directories are redundant. Closing of files always progresses in the correct order so that when the user is notified that a file is closed, it is known to be safe.

All file system write operations are ordered so that the directions are always correct and are left in a consistent state. When operations on a file are active, index blocks and pointers are kept in core for speed.

File and process space is integrated providing added convenience and security. This integration eliminates race conditions in transaction processing and data base applications, because unlike conventional systems, when a file is updated it is the file itself, and not a copy that is modified.

Disk and file space is dynamically allocated. There is no requirement to preallocate disk space. A complete utility is provided for backing up or dumping files. All variations of file backup procedure are available—full, incremental, and selected directories. Provision is made for implementation of a clocked dumping procedure.

Named Files

The File System on the DECSYSTEM-20 is a collection of named data sets (files) divided into 512 word pages. Each file is identified by a device, directory name, file name, and extension and generation, which defines a unique path to the file.

File Protection

All files are assigned protection codes for each of three classes of users: the owner, users within a common group or project, and all other system users. Members of each class may be granted none, some, or all of the following privileges:

- Read access
- Write access
- Permission to append data to the end of the file
- Acknowledgment that the file exists when listing the directory

When a file is created, the creator/owner may either expli-

citly assign a protection code or take the system default protection. The owner is permitted to change the protection code at any time.

Each file has associated with it a File Descriptor Block which fully describes the file's characteristics to the system. Such information includes its protection code, who the owner is, who last wrote in it, the date created, the date last changed, and its size.

File Directories

The system maintains a file directory for each user. It contains pointers to all of the user's files and file descriptor blocks as well as general information about the user. This information includes the user's password, his privileges, his disk space quota, how much disk space he has used, and the system default protection code. All directories are themselves named files, and information about them is contained in a master directory called the Root Directory. There are redundant Root Directories to increase file system integrity.

Groups

A group is a set of cooperating users established by the system administrator. Each directory contains two group membership lists: the list of users that may have group level access to the files in this Directory and the list of groups to which the owner of this Directory belongs.

Instructor/Student Example

A way in which controlled access may be used is illustrated by a typical instructor/student situation. The instructor is a member of Group A as are other instructors, but his directory may not be accessed by Group A. On the other hand, student directories are all accessible by Group A so that the instructor and all other instructors can have group level access to student files at any time, but neither students nor other instructors may access the instructor's files.

To allow for this controlled file sharing, the system offers several modes of shared access:

- Shared reading (one or more users reading, none writing)
- Writing (one user writing, none reading)
- Shared updating (one or more users reading and writing simultaneously, using the DECSYSTEM-20 queuing facility for coordination)
- Restricted updating (one user writing, any number reading)

As a further protection, if a user has opened a file for reading only, or is executing a shared program, and attempts to write in a shared page, the user who wishes to write is given a private copy of the page and the write proceeds. This "copy-on-write" facility is automatic.

All the software and hardware elements associated with the file system have been designed to provide controlled sharing of files among users on the system. Pages of core memory function as a cache between the file system and the processor.

File Backup

A system utility DUMPER provides full file backup capabilities. Capable of full or incremental file saving with provision for directory specification, DUMPER is a complete utility for saving and retrieving files.

Virtual Memory Operations

The TOPS-20 Operating System provides a demand-paged virtual memory environment. Each job on the system can be considered as a set of one or more processes, each having a unique 256K word addressing space and linked together by user options or system-defined linkages for efficient use of storage and systems resources. Operating in a demand-paged mode, the system will move into memory a subset of the total number of pages of each active process, bringing in pages from the disk or moving pages to the disk using algorithms based on the collective characteristics of all processes active on the system. A primary property of every process is its Working Set, defined as the set of pages referenced during a recent interval of time. The system determines the interval on the basis that when next the process is active, memory references will be confined with reasonable probability to the same pages. The number of process pages comprising the working set is determined by program characteristics.

A second property, definable in terms of all active processes on the system, is the Balance Set, which is the set of processes

most eligible to run and whose collective Work Sets will fit into core. Process eligibility is event driven and changes when the state of the process changes.

To insure good response, interactive processes merit a higher run eligibility than computation-bound processes. However, if any compute-bound process has been blocked by interactive processes for a long period, its eligibility is changed so as to allow it to run.

The scheduler periodically defines the Balance Set. Working Set sizes are modified dynamically as a process runs and are regularly monitored by the scheduler to adjust Balance Set membership. Administrative controls exist to guarantee a percentage of the processor to a specific user.

All paging is on a demand basis. A recently started process may begin with no pages in core. As it makes memory references, page faults occur (temporary failures due to referencing pages which are not in core) and a Working Set is built up. File input and output is accomplished by demand paging as well; the desired page is mapped into the system or user address space and is then referenced, causing a page fault.

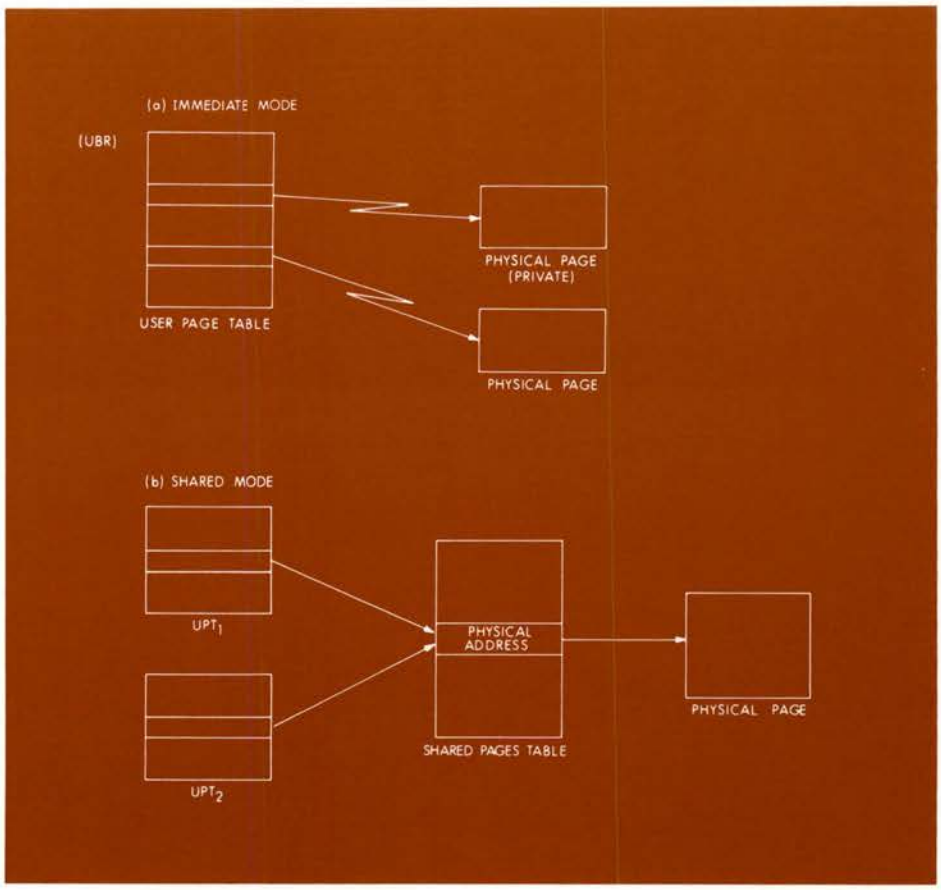
The DECSYSTEM-20 supports memory management in several important ways through hardware and microcode. Memory mapping and page level access protection are provided, so as to provide efficient sharing of pages between processes and efficient context switching. Page status and age information are automatically updated for each in core page by the microcode from information initially set up by the software. These hardware tables minimize overhead while maximizing the amount and accuracy of information available to the system for decision making in its management of core and minimization of disk channel traffic.

To effectively manage memory, the DECSYSTEM-20 uses a combination of hardware and software registers and tables to relate program or virtual addresses to physical addresses. Two hardware registers internal to the Central Processor, the EBR and UBR, contain pointers which locate the physical pages in memory containing the mapping information for the operating system and the currently active user. These pages, named the User Page Table and the Monitor Page Table, contain pointers for mapping information between the user's and monitor's address space and the actual pages of physical memory in use by the user and operating system. The pointers contained in the process tables are of three distinct types:

Immediate
Shared
Indirect

The characteristics of an Immediate Mode pointer are shown below. (a)

Entries in the User Page Table point directly to physical pages in core. Shared Mode Pointer characteristics are shown below. (b)



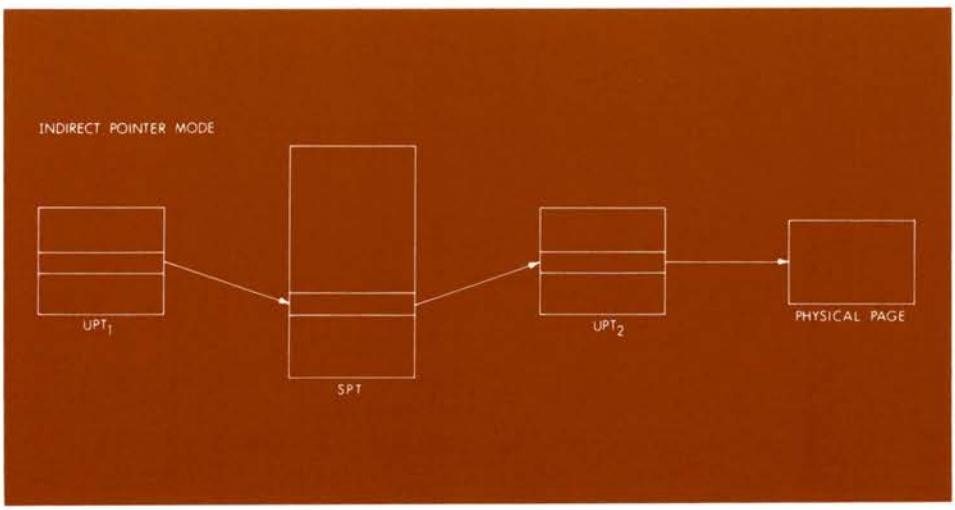
In this case, the pointer indexes into a Shared Pages Table, whose entries point to physical pages in memory. The location of the Shared Pages Table in memory is contained in another hardware register internal to the Central Processor. The advantage of having a Shared Page Table occurs when multiple processes are sharing the same page. The bookkeeping required to track the location of the shared page is reduced to updating only the Shared Page Table. Without a Shared Page Table, every user's Page Table, many of which would not be in core, would have to be modified. In instances where one process writes on a shared page, the system performs a "Copy-on-Write" action, resulting in a separate copy of page being created for the user, containing his modification. An appropriate update is then made to the information maintained in the Shared Pages Table to reduce by one the number of users sharing the specific page.

Indirect Pointer Mode is shown here:

In this mode, the entry on the Shared Pages Table does not point directly to memory, but rather points to another entry in the Process Table, which itself may re-index back into the Shared Pages Table or point directly to a physical page.

To maximize system speed, a subset of the information for all active process page mappings, including the operating system, is kept in a Hardware Page Table. As the Working Set for a particular process is generated or altered, the page mappings obtained are loaded into the Hardware Page Table. Subsequent memory reference to these pages then proceeds through the Hardware Page Table, eliminating the need to access the in-core page tables. This high-speed hardware page table minimizes the need for an "extra" cycle to pick up the required mapping information.

One additional register internal to the Central Processor points to a Core Status Table, which contains, for each page of physical core, information on recent references, page modifications, reasons for a page to be loaded in core, and reverse pointer data. The Core Status Table is used by the operating system to age pages, record when pages have been written into, record which processes have referenced the page, and to permit the operating system to locate which of the page tables point to the page.

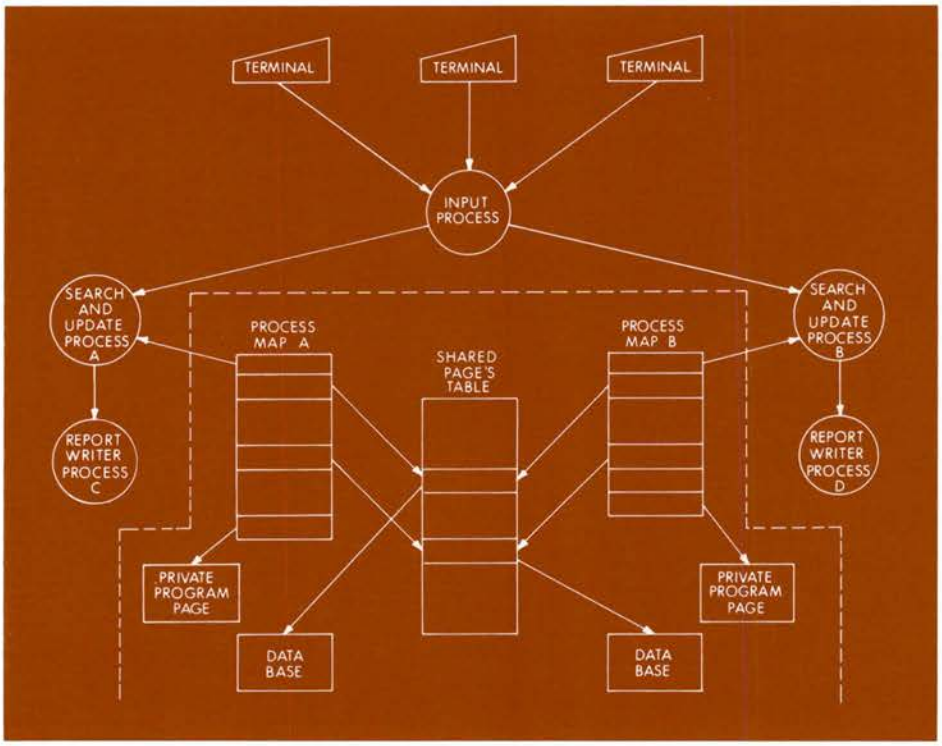


Process Structure

A process, as defined for the DECSYSTEM-20, is an independent task capable of being run by the monitor and having its own environment, including a 256K virtual address space divided into 512 pages of 512 words each. Core management is performed by the operating system, and only active process pages need be in memory for execution.

A process may create other processes which are inferior to the creating process. Two processes which have been created by another process are called parallel processes. The superior, or creating, process has control over its inferior processes and may confer or withhold privileges and suspend, continue, or terminate them.

The simplest example of a process is a single task, such as a program to solve a specific problem. It receives data, performs calculations, outputs an answer, and then exits. More complicated examples may have one task receive inputs from terminals, and pass them to subtasks for processing. These processes, in turn, may initiate other tasks to perform specific functions. The following diagram illustrates such an application.



Because each process is scheduled by the operating system individually, the system can be more responsive to the demands of the application. The ability of the system to perform a variety of functions apparently simultaneously is more easily achieved. Response time to the input terminals may be enhanced since the top level process need only receive input requests and need not be delayed due to what in another design might be a delay to update a data base.

Interprocess Communication Facility (IPCF)

Although each process is fundamentally independent, an important advantage of a multiprocess job structure is that output of some processes can be taken as input to others. The Interprocess Communication Facility allows for fast communication among processes. This communication occurs when processes send and receive information in the form of packets (messages). Each sender and receiver has a unique Process I.D. assigned to it, which is used to route packets between processes.

When one process sends information to another, it is placed into the receiver's input queue (a shared page) where it remains until the receiver retrieves it. Instead of periodically checking its input queue, the receiver can enable the software interrupt system so it can be interrupted by the sending process when the information has been sent.

To use IPCF, the system administrator must assign a user two quotas which designate the number of sends and receives his process may have outstanding at any time. For example, if a user has a send quota of two and has sent two messages, he cannot send any more until at least one has been retrieved by its receiver. A user cannot use the facility if his quotas are zero.

Software Interrupt System

The software interrupt system provides a mechanism for a process to respond to several types of interrupts generated by other processes, hardware, error conditions, and terminal interactions. It may be enabled by both the user and the system. The system provides 36 software interrupt channels, of which 18 have reserved functions. The primary types of interrupts are:

- Terminal character interrupts
- IPCF interrupts
- ENQ/DEQ interrupts
- Hardware error conditions interrupts
- I/O error condition interrupts
- Program errors, such as arithmetic overflow
- Interprocess interrupts

There are three interrupt priority levels assignable to each software interrupt channel, thus providing priority ordering. A call is provided for interrupt

dismissal to resume the interrupted code.

Enqueue/Dequeue (ENQ/DEQ)

Although TOPS-20 makes program and data sharing very easy, while preventing any user from changing a file which others are using in read-only mode, more complex applications require simultaneous updating of a data base (file) or sharing of devices.

By using ENQ/DEQ facility, cooperating processes can insure that such resources are shared correctly and that one user's modifications do not interfere with another's. Examples of resources that can be controlled by this facility are devices, files, operations on files (e.g., READ, WRITE), records, and memory pages.

Front End Software

The Front End Processor and software reduce the central processor's participation in I/O operations and provide an extensive diagnostic/maintenance tool.

The purpose of the Front End Processor is to reduce the workload of the Central Processor and to serve as a powerful diagnostic and maintenance tool for service personnel. The following functions are carried out by the Front End Processor:

- Console processor
- Communications processor
- Peripheral processor
- Diagnostic/Maintenance processor

Console Processor—Two major functions of the console processor are system initialization and system-operator communications. A person need only push a button and type in the date to initiate the following automatic program sequence. The Front End Processor loads and verifies the microcode, configures and interleaves memory, and loads and starts execution of the Central Processor bootstrap program from a device specified by the user.

The user may request a memory configuration list indicating which memory controller is on line, what is the highest memory address configured, and how the memory is interleaved.

All normal console capabilities, such as lights, start, stop, reset, and load, are provided by the Front End Processor. An operator command language is provided with the system.

Communications Processor—The Front End Processor serves as an intelligent data link and buffer for the interactive terminals, thereby relieving the Central Processor of this overhead. The Front End Processor buffers the characters received and interrupts the Central Processor upon a clock signal or when it has a group of characters to send. The Central Processor also sends groups of characters for terminal output to the Front End Processor, further enhancing efficiency. Programmable terminal speed setting capability is provided, allowing for terminal speeds to be changed dynamically.

Peripheral Processor—The unit record spooling programs in TOPS-20 communicate with the Front End Processor by the same buffering mechanism described for the communication processor. Both the Central Processor and the Front End Processor maintain buffers for data, and interrupt only once per buffer transmission, thus increasing the amount of useful work each processor can do.

Diagnostic/Maintenance Processor—An integral part of the DECSYSTEM-20 is a remote diagnosis capability that reduces Mean Time To Repair (MTTR) and increases system availability to the customer.

This capability allows Service Engineers to determine the cause of most system failures before leaving the local office. By running diagnostic tests using telephone dial-up facilities, the Field Service Engineer can give the customer better service because he can better select which spares and test equipment should accompany him on the call. Most tests may be run during general timesharing. Even if the main CPU is inoperative, the Front End Processor can access all busses and major registers.

Total system security is maintained because the on-site system operator must enable the communications link, and only the on-site personnel may specify the specific password to the system each time remote diagnosis is employed. Time limits for system access may also be imposed, and the remote link may never operate at a higher privilege level for commands than the local console terminal. All input and output to the diagnostic link is copied to the local terminal so that on-site personnel have a record of all steps taken.

Languages

System Reporting and Control

One of the major system design goals for the DECSYSTEM-20 has been to provide a high degree of control and simplicity of operation. Areas of specific concern include:

Accounting and Administration—Files are maintained by the system to provide information on resource usage (CPU, disk space, etc). This provides the basis for charging individual users as well as providing a year-to-date summary and usage statistics.

The system administrator sets resource allocation guidelines: disk quotas for each individual user to control disk usage, the amount of resources for batch and timesharing, and the maximum sizes of spooled output files.

Certain privileges can be granted to key personnel. These could cause an unnecessary loss of system security and reliability if used incorrectly. To further protect the system and the privileged user, his privileges are only in effect if he specifically enables them.

System Generation—The monitor is self-configuring.

Error Reporting—The TOPS-20 Operating System incorporates an extensive error detection and recovery package to insure maximum system availability to the user. It also incorporates complete error recording facilities for use by DIGITAL maintenance engineers and support personnel, and customer operating staff.

When an error is detected, the TOPS-20 monitor gathers all pertinent hardware and software information. The recovery procedure is then started and error related information is added to a disk file for later use.

The DECSYSTEM-20 is provided with an extensive range of user oriented high-level languages. The development of high-level language has been an important process in the evolution of improved computer systems.

The solution to problems can best be expressed in the terms most closely related to the problem. The scientific user has come to depend upon FORTRAN, ALGOL and APL. Commercial users apply COBOL to business data processing tasks.

For the beginning or casual user, BASIC is often the first choice. Interestingly, many useful and extensive applications packages have been developed in the BASIC language.

Just having language translators is not sufficient. The DECSYSTEM-20 user's work has been made easier by providing extensions to standard languages and by relaxing certain restrictions.

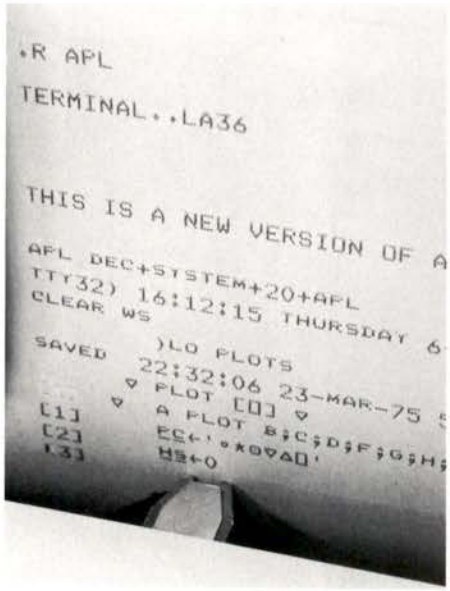
Better language implementations permit the computer to handle more of the annoying details while the user can spend more of his time solving problems.

Because writing a program is only one step in the program development process, the DECSYSTEM-20 provides not only a superior interactive facility for entering the program, but also, superior debugging tools which operate at the source program level.

DIGITAL originated the interactive debugging concept when it was first to introduce DDT, (Dynamic Debugging Technique) variants of which are now offered on other systems. To make the usefulness of interactive debugging available to a wider range of people, the DECSYSTEM-20 FORTRAN and COBOL have source level debugging facilities—FORDDT and COBDDT.

These tools help to maximize the user's efficiency through the assistance they provide in getting his program operational.

To ensure that programs run as efficiently as possible, DECSYSTEM-20 produces efficient code taking advantage of the program's structure and the superior instruction set of the DECSYSTEM-20. To minimize core requirements DECSYSTEM-20 compilers are re-entrant and may produce re-entrant code.



ALGOL

- Programs may comprise separately-compiled procedures
- Assignments are permitted within expressions
- Remainder operator
- Unique implementation of dynamic own arrays
- Octal boolean constants and integer/boolean and boolean/integer transfer functions
- Alternative reserved delimiter word or "quoted delimiter word" representations

The DECSYSTEM-20 ALGOL system is based on the ALGOL-60 specifications. It is composed of the ALGOL compiler, and the ALGOL object time system. The compiler is responsible for reading programs written in the ALGOL language and converting these programs into machine language. The user may specify a parameter at compile time which produces a sequence numbered listing with cross-referenced symbol tables.

The ALGOL object time system provides special services, including the input/output service for the compiled ALGOL program. Part of the object time system ALGOL library is a set of routines that the user's program can call in order to perform various functions including mathematical functions and string and data transmission routines. These routines are loaded with the user's program when required; the user need only make a call to them. The remainder of the object time system is responsible for the running of the program and providing services for system resources, such as core allocation and management and assignment of peripheral devices.

Source level, interactive debugging is provided through ALGDDT. The user may stop his program at any point, may examine and change the contents of data locations, may insert connected code and then continue execution to test his changes.

APL

APL (A Programming Language) is a concise programming language specially suited for dealing with numeric and character data which can be collected into lists and arrays. The conciseness with which APL expressions can be written greatly enhances programmer productivity and allows for very compact and readable code which can be executed in a highly efficient manner. APL finds application not only in mathematics and engineering, but also in financial modeling and text handling situations.

Some of APL's features include:

- The user's active workspace size is dynamically variable
- The workspace symbol-table is dynamically variable
- Immediate-mode line editing of complex APL expressions
- Statement branching may occur anywhere in a statement line
- Multiple statements may appear in a single line
- User-controlled tab positioning for I/O operations
- All floating point operations are performed to an accuracy of eighteen decimal digits

APL is a fully interactive system with both immediate (desk calculator) and function (program) modes of operation. It includes its own editor as well as debugging tools, tracing of function execution, typeout of intermediate values, setting break points, etc. APL for the DECSYSTEM-20 includes many extensions not normally found in other APL implementations. Some of these features include a flexible file system which supports both ASCII sequential and binary direct access files, the dyadic format operator, expanded command formations which allow the user to take advantage of the DECSYSTEM-20 file system, the execute operator and tools for error analysis and recovery. In addition, workspaces are dynamically variable in sizes up to 512K bytes and the system supports double precision calculations which allow up to 18 decimal digits of precision.

Finally, APL on the DECSYSTEM-20 is available to all users at all times without the need to run a separate subsystem which provides extra terminal handling, workspace swapping, etc. These tasks are all accomplished by the TOPS-20 operating system.

BASIC+2

- Multiple users share the BASIC compiler's pure code
- A variety of program manipulation commands which include functions for saving, running and retrieving BASIC programs
- Immediate mode statements to facilitate debugging and "desk calculator" mode
- Automatic syntax checking
- Advanced editor
- 30 character variable names
- Sequential data storage
- String capability; including string arrays and functions
- Chaining with COMMON to accommodate large programs
- Formatted output using the PRINT USING statement

BASIC is a conversational problem solving language that is well suited for timesharing and easy to learn. It has wide application in the scientific, business, and educational communities and can be used to solve both simple and complex mathematical problems from the user's terminal.

The BASIC user types in computational procedures as a series of numbered statements that are composed of common English terms and standard mathematical notation. After the statements are entered, a run-type command initiates the execution of the program and returns the results.

If there are errors during execution, the user, from his terminal, simply changes the line or lines in error by deleting, modifying, or inserting lines.

The beginning user has many facilities at his disposal to aid in program creation:

- **Program Editing Facilities**—An existing program or data file can be edited by adding or deleting lines doing context searches, making local or global changes, renaming it, or by resequencing the line numbers. The user can combine two programs or data files into one and request either a listing of all or part of it on the terminal or a listing of all of it on the high-speed line printer.
- **Documentation Aids**—Documenting programs by the insertion of remarks within procedures enables recall of needed information at some later date and is invaluable in situations in which the program is shared by other users.
- As a BASIC user, you can type in a computational procedure as a series of numbered statements by using simple common English syntax and familiar mathematical notation. You can solve almost any problem by spending an hour or so learning the necessary elementary commands.
- **Functions**—Occasionally, you may want to calculate a function, for example, the square of a number. Instead of writing a program to calculate this function, BASIC provides functions as part of the language.

More advanced users will want to take advantage of some of the more sophisticated computation and data handling tools which BASIC provides. For example:

File Input—The user may create a name date file using the BASIC editing facilities. This file may now be referenced within a program.

Output Formatting—The user can control the appearance of his output to the terminal or printer.

Data Access—Data files may be read and written either sequentially or randomly.

String Manipulation—Alphanumeric strings may be read, printed, concatenated and searched.

Error Handling—The user controls the action taken under program control when an error condition occurs via the CM ERROR statement.

COBOL

The COmmon Business Oriented Language, COBOL, is an industry-wide data processing language that is designed for business applications, such as payroll, inventory control, and accounts receivable.

Because COBOL programs are written in terms that are familiar to the business user, he can easily describe the formats of his data and the processing to be performed in simple English-like statements. Therefore, programmer training is minimal, COBOL programs are self-documenting, and programming of desired applications is accomplished quickly and easily.

Major features include:

- **On-line editing and debugging**
  The programmer may create and modify the source program from a terminal using an easily-learned editing facility. After the program has been submitted to the compiler—again from the terminal—any syntax errors may be immediately corrected using the editor, and the program resubmitted. The program listing is sequence numbered with symbols cross-referenced to show when they are defined and where used. COBDDT, the on-line, interactive COBOL debugging package, allows the programmer to check out the program:
  - By selectively displaying a paragraph name
  - By causing the program to pause at any desired step during execution
  - By allowing the program to examine and modify data at will before continuing execution

  Easy program development and debugging increases programmer productivity by eliminating tedious waiting periods.
- **Batch**
  Using the same commands for timesharing, the programmer may submit the program via a control file to the Batch system. This further increases efficiency because other terminal work may be done concurrent with the Batch processing
- **Access Methods**
  The programmer has a choice of three access methods; sequential, indexed sequential, and random.

Indexed Sequential Access Mode (ISAM)

ISAM requires a minimum amount of programming while providing a large data file handling capacity. It is supported by the COBOL Object Time System which automatically handles all of the searching and movement of data.

All reading and writing of an index file (ten levels of indexing) are performed by the run-time operating system (LIBOL). This does not involve the user. When using the indexed sequential files, the programmer need only specify which record is to be read, written, rewritten, or deleted.

Whenever records are added to the file, the index is automatically updated, additions to the file will not degenerate the file as with other computers. The common technique of using overflow areas for added records has been avoided. Whenever records have been deleted from the file, the empty space is used again for later additions. The net effect of the addition and deletion techniques significantly increases the time between major "overhauls" of the data files, because the time required to access a file is independent of the number of changes made to it.

- **Sorting**
  The SORT package permits a user to rearrange records or data according to a set of user-specified keys. The keys may be ascending or descending, alpha or numeric, and any size or location within the record. Multi-reel file devices may be specified.
- **Source Library Maintenance System**
  This system lists file entries or adds, replaces, and/or deletes source language data on a file. It can also add, replace, or extract an entire file from the library.
- **Device Independence**
  The operating system allows the programmer to reference a device with a user-assigned logical name as well as its physical name, thus allowing dynamic assignment of peripheral devices at run time.
- **Data Base Management System Interface**
  The programmer may call upon the data base management system facilities from within his program. This system, which follows the CODASYL specifications, allows data files to be consolidated into one or more data bases. Application programs are then permitted to access the data in the way best suited to their needs.

CPL (Conversational Programming Language)

CPL is an interpreter supporting a subset of the draft ANSI PL/I language. The following is a summary of the key features of CPL:

- Data types include FIXED, FLOAT, CHARACTER, CHARACTER VARYING, BIT, BIT VARYING, POINTER, and arrays of these data types.
- Storage classes include AUTOMATIC, STATIC, CONTROLLED and BASED
- Almost all PL/I statement types are supported including READ, WRITE, DECLARE, DEFAULT, GET, PUT, ON and FORMAT
- Subroutines and function procedures are recursive
- The ON statement provides the capability of recovering from program error under program control.
- Complete string manipulation package
- CPL supports the following classes of PL/I built-in functions: arithmetic, mathematical, string array, storage control and pseudo-variables.

CPL is intended to be easy-to-use for the beginning programmer or non-programmer. At the user's option, statements will be executed immediately or saved for deferred execution. A beginning programmer can start by executing simple computational statements and can proceed to building programs. Since CPL is an interpreter, a user can track a program very closely. Debugging features, such as source level breakpoints and program modification are available.

FORTRAN

The FORmula TRANslator language, FORTRAN, is a widely-used and procedure-oriented programming language. It is designed for solving scientific problems and is thus composed of mathematical-like statements constructed in accordance with precisely formulated rules. Therefore, programs written in FORTRAN consist of meaningful sequences of these statements that are intended to direct the computer to perform the specified computations.

FORTRAN has a wide use in every segment of the computer market. Universities find that FORTRAN is a good language with which to teach students how to solve problems via the computer. The scientific community relies on FORTRAN because of the ease with which scientific problems can be expressed. In addition, FORTRAN is used as the primary data processing language by many timesharing utilities.

FORTRAN Version 4

FORTRAN-20 is a superset of the American National Standard FORTRAN. Both the compiler and object-time system are re-entrant (shareable). The compiler produces optimized object code. The following is a summary of key features and extensions of FORTRAN-20:

- PARAMETER statement—Allows symbolic specification of compile-time constants
- INCLUDE statement—Allows user to include in the compilation of a given program unit source code which resides on a file other than the primary source file
- GLOBAL OPTIMIZATION-FORTRAN-20 performs extensive local and optional global optimizations
- FORDDT—the FORTRAN-20 debugger
  FORDDT in conjunction with the FORTRAN-20/"DEBUG" switch will allow:
  - Display and modification of program data
  - Tracing of the program statement by statement
  - Setting of pauses on any statement or routine
- OPEN/CLOSE file-specification statements
- Array bounds checking can be generated optionally
- N-dimensional arrays
- ENCODE/DECODE statements
- Boolean operations equivalence (EQV) and exclusive or (XOR), in addition to OR, AND, NOT.
- The NAMELIST and list-directed I/O features provide format-free input and output operations
- Random-access I/O capabilities
- Compatibility with IBM type declaration statements
- Implied DO loops in I/O statements and data statements
- Full mixed-mode arithmetic in expressions
- Octal Constants

- Logical Operations—full-word, masking operations for all logic functions (rather than a result of just true or false)
- END= and ERR= in I/O statements
- Device independence

The FORTRAN-20 object-time system, FOROTS, controls the input/output, format interpretation and numerical conversion for compiled programs. The FORTRAN user may reference any I/O device. All special editing, conversion, and file structuring tasks are handled by the object-time system. Devices are normally specified by logical assignment so that physical device selection need not be made until run-time. The devices corresponding to the special I/O statements READ, PRINT, ACCEPT, and TYPE are also assignable at run-time. The object-time system is shareable.

FOROTS implements all program data file functions and provides the user with an extensive run-time error reporting system. Device independence is provided to allow the programmer or operator to determine the physical device at run time.

FORTRAN-20 is easy to use in both timesharing and batch processing environments. Under timesharing, the user operates in an interactive editing and debugging environment. FORDDT, an interactive program that is used as an aid in debugging FORTRAN programs uses the language constructs and variable names of the program itself—thereby eliminating the need to learn another language in order to accomplish on-line debugging. Under batch processing, the user submits his program through the batch software in order to have the compiling, loading, and executing phases performed without his intervention.

FORTRAN programs can be entered into the FORTRAN system from a number of devices: disk, magnetic tape, user terminal and card reader. In addition to data files created by FORTRAN, the user can submit data files or FORTRAN source files created by the system editor. The data files contain the data needed by the user's object program during execution. The source files contain the FORTRAN source text to be compiled by the FORTRAN compiler. Output may be received on the user's terminal, disk or magnetic tape. The source listing cross references all symbols to show where they are defined and where used.

MACRO

MACRO is the symbolic assembly language on the DECSYSTEM-20. It makes machine language programming easier and faster for the user by:

- Translating symbolic operacodes in the source program into the binary codes needed in machine language instructions
- Relating symbols specified by the user to stored addresses or numeric values
- Assigning relative core addresses to symbolic addresses of program instructions and data
- Providing a sequence numbered listing with symbols cross-referenced to show where they are defined and where used

MACRO programs consist of a series of free format statements that can be prepared on the user's terminal with a system editing program. The elements in each statement can be entered in free format. The assembler interprets and processes these statements, generates binary instructions or data words, and processes a listing which may contain cross reference symbols for ease in debugging. MACRO is a device-independent program; it allows the user to select, at run-time, standard peripheral devices for input and output files. For example, input of the source program can come from the user's terminal and output of the assembled binary program can go to a magnetic tape, and output of the program listing can go to the line printer. More commonly, the source program input and the binary output are disk files.

The MACRO assembler contains powerful macro capabilities that allow the user to create new language elements. This capability is useful when a sequence of code is used several times with only certain arguments changed. The code sequence is defined with dummy arguments as a macro instruction. Thus, a single statement in the source program referring to the macro by name, along with a list of the real arguments, generates the

entire sequence needed. This capability allows for the expansion and adaptation of the assembler in order to perform specialized functions for each programming job.

DDT

The on-line symbolic debugging tool, DDT (Dynamic Debugging Technique), enables the user to perform rapid checkout of a new program by making a change resulting from an error detected using DDT and then immediately executing that section of the program for testing and loading; the user may enter DDT at any time, either before execution or after an error has occurred. After the source program has been assembled, the binary object program, with its table of defined symbols, is loaded with DDT. Through command strings to DDT, the user can specify locations in his program, called breakpoints, where DDT is to suspend execution in order to accept further commands. In this way, the user can check out his program section-by-section and, if an error occurs, insert the corrected code immediately. Either before DDT begins execution or at breakpoints, the user can examine and modify the contents of numbered text modes. DDT also performs searches, gives conditional dumps, and calls user-coded debugging subroutines at breakpoint locations. The important feature of DDT is that user communication with DDT is in terms of the original symbolic location names, rather than the machine-assigned locations.

EDIT

Edit allows on-line creation and modification of programs and data files. The user may insert, delete, or print lines, as well as modify lines without having to retype them. Advanced techniques include string searches, substitutions and text manipulations.

The novice finds the system easy to use due to the user-oriented syntax and extensive helping facilities. At the same time the more advanced user may perform quite intricate text manipulation with equal ease.

DUMPER

Dumper provides file backup for the disk file structure. Installations may use it to dump and restore all files in all directories, to dump and restore only those files which have been changed since last dump, and to dump and restore individual user directories.

LINK

LINK, the DECSYSTEM-20 linking loader, merges independently-translated modules of the user's program and any necessary system modules into a single module that can be excuted by the operating system.

The primary output of LINK is the executable version of the user's program. The user can also request auxiliary output in the form of map, log, save, symbol, overlay plot, and expanded core image files by including appropriate instructions in his command strings to LINK. The user can also gain precise control over the loading process by setting various loading parameters and by controlling the loading of symbols and modules. Furthermore, by setting parameters in his command strings to LINK, the user may specify the core sizes and starting addresses of modules, the size of the symbol table, the segment into which the symbol table is placed, the messages he will see on his terminal or in his log file, and the severity and detail of any error messages. Finally, he can accept the LINK defaults for items in a file specification or he can set his own defaults that will be used automatically when he omits an item from his command string.

RUNOFF

RUNOFF is the DECSYSTEM-20 documentation preparation program. It provides line justification, page numbering, titling, indexing, formatting, and case shifting. The user creates a file which includes text and information for formatting and case shifting, with an editor. RUNOFF processes the file and produces the final formatted file to be output to the terminal, the line printer, or to another file.

With RUNOFF, large amounts of material can be inserted into or deleted from the file without retyping text that remains unchanged. After a group of modifications has been added to a file, RUNOFF produces a new copy of the file which is properly paged and formatted.

SORT

The TOPS-20 SORT arranges the records of one or more files according to a user-specified sequence. The user designates the keys on which the records are sorted from one or more fields within a record. The keys can be in either ascending or descending order. SORT compares the key fields values of all records. Then it arranges the records in the specified sequence and merges them into a single output file. Sort may be used under timesharing and batch, and may be called from within a COBOL program.

Data Base Management System

Certain data—such as that within commercial, accounting, inventory control and administrative systems—are used in computer applications which have common relationships and processing requirements with data in other applications. This can prove to be a problem because as file organizations are defined in one application or program, restructuring, redundant appearance, and even repetitive processing of the same data are often required in another application.

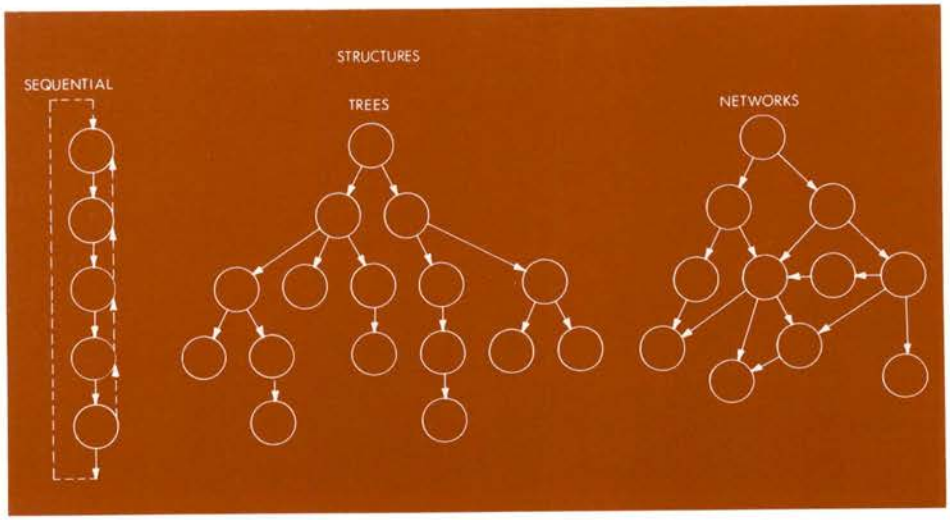
To further complicate the problem, on-line processes (i.e., customer order entry, shipment planning, and student information retrieval systems) tend to different data structures than the processes used to create and maintain primary data files. This means that as new applications requirements for existing data are determined and additional data are defined, program development personnel must either alter existing data file forms and programs or create and maintain redundant copies.

DIGITAL offers a solution to the problem in the Data Base Management System—DBMS. It enables DECSYSTEM-20 users to organize and maintain data in forms more suitable to the integration of a number of related but separate processes and applications. DBMS is ideal in situations where data processing control and program development functions require structures and techniques not satisfied by traditional data management facilities.

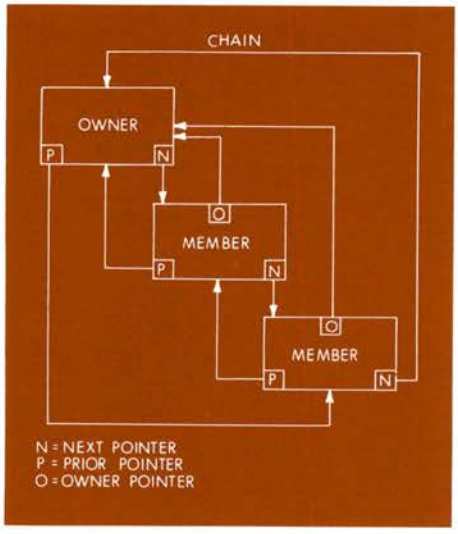
Features

The DECSYSTEM-20 Data Base Management System is predicated upon the proposals of the CODASYL Data Base Task Group (DBTG) which appear in their report of April, 1971. DIGITAL's goal for DBMS is to provide features which assist users in obtaining the most significant objectives stated in the CODASYL DBTG report. These features include:

- **Hierarchical Data Structures**—In addition to sequential structures, simple tree structures and more complex network structures can be created and maintained. Data items can be related within and between various levels of the structure established.



- **Non-redundant Data Occurrence**— Data items may appear in a number of different structural relationships without requiring multiple copies of the data. Data structures may be established and modified in a manner most suitable to a given application without altering the occurrence of the data in structures maintained for other applications.
- **Variety of Access and Search Strategies**— Data records can be maintained in chains, as illustrated below. Access may be through DIRECT, CALCULATED, or VIA set location modes. Sort keys, in addition to the normal storage key, may be defined.



- **Concurrent Access**— Multiple run units (or programs) that use the same re-entrant code module further supplement the DECSYSTEM-20 monitor's extensive centralized file handling capabilities. Any number of concurrent retrievals to the same data areas can be handled. Concurrent updates to the same area can be handled by a multiple update queuing mechanism.
- **Protection and Centralized Control**— Usage of a common data definition language establishes data base structures maintained on direct access storage devices that are selectively referenced by individual application programs through privacy lock and key mechanisms. Physical placement of data is specified by a centrally-controlled data definition processor.
- **Device Independence**— The common input/output and control conventions of the DECSYSTEM-20 monitor provide basic device independence. Applications programs deal with logical areas rather than physical devices. Data base areas may reside on the same or different direct access storage devices as non-date base files.
- **Program Independence of Data**— Significant steps toward program independence are achieved by using the SCHEMA and SUB-SCHEMA concepts of data definition. Individual programs reference only user-selected data elements rather than entire record formats. Program alterations due to adding new data and relationships are minimized. Alterations of the original form (i.e., binary, display) and element sizes require program recompilations.
- **User Interaction Without Structural Maintenance Responsibilities**— Individual users, applications, and programs may access data structures without the responsibility of maintaining detailed linkage mechanisms that are internal to the data base software. Common and centralized authorization, error recovery, and building techniques reduce individual activity to maintain data structures.
- **Multiple Language Usage**— The DBMS data manipulation language is available to both COBOL and FORTRAN as host languages
- DBMS Software Modules Consistent with the CODASYL Data Base Task Group report, the body of DECSYSTEM-20 data base management software includes:
  - DDL—data description language and its processor
  - DML—data manipulation language for COBOL and FORTRAN programs
  - DCBS—data base manager module of re-entrant run-time routines
  - DBU—group of data base system support utilities

Services

Quality in DIGITAL computer products is acknowledged throughout the world. Our engineers and programmers have provided reliable computer hardware and software to solve a host of problems in a variety of fields. To complement our product offering, we have developed a comprehensive customer service capability.

Over the past few years, Datamation Magazine has run an annual survey of computer users, asking their opinions on the quality of service they receive from their vendors. One major area of concern is post-installation maintenance services. DIGITAL has continually ranked first in quality and availability of service.

Major factors in this capability are the number and placement of service personnel; parts inventories available to support local efforts; an organizational structure that provides increasing expertise at each level; a management reporting system that continuously monitors all areas of service agreements to meet every customer need.

People

DIGITAL's hardware maintenance organization includes 3000 hardware specialists supported by an administrative staff of over 1000. These specialists are located in more than 300 worldwide offices for the greatest decentralization of service availability.

Large Systems service engineers typically enter DIGITAL with more than four years of field experience. They then undergo four months of extensive training at DIGITAL, followed by yearly updates in new products and technology. They work with product support teams who assure smooth transitions of new products—from development to testing to installation—and provide consultation and assistance for all service locations.

Parts

The size and location of DIGITAL's parts inventories—at local, district, regional, and headquarters operations—are computer-optimized by a DECsystem-10, which takes into account the needs of both existing and new installations. The DECsystem-10, in concert with PDP-11 computers at regional offices, assures that parts resources are available locally to solve more than 90% of user problems.

For those problems requiring parts not available immediately, DIGITAL has a priority parts system utilizing inventories at district, regional and corporate levels. All priority parts are shipped by specialized air-freight carriers.

Support

Field Service's hierarchical organization places the greatest concentration of resources at the local level, with supporting facilities at each ascending level in the group. Continuous monitoring of service activity at all levels of Field Service management focuses attention on our worldwide systems base so that resources can be mobilized as required.

Service Agreements

To give flexibility to this structure and provide services that meet users exact needs, DIGITAL offers a variety of service agreements. These agreements range from coverage 12 hours/day, five days/week to around-the-clock maintenance coverage. Occasional service, on a time and materials basis, is available to users who elect to perform self-maintenance, as well as off-site repair of electro-mechanical assemblies through 16 Product Repair Centers in the U.S. and in Europe.

Logistics

Good service demands outstanding logistics. Spares availability minimize computer repair time. Our materials inventory network recognizes this concept—and strengthens the total DIGITAL support system.

The first level of spares is often at your own computer site. From this point the inventory is in echelons at the Branch, District, Regional and Headquarters locations. Our own DECSYSTEM-20 keeps track of this support system, providing replacement parts when needed.

For DIGITAL computers located in areas on the fringes of our service capabilities, and for customers who use our maintenance on an "as available" basis, DIGITAL supplies spares planning and inventory procurement assistance.

DECSYSTEM 20 Training

Education

Education is an integral part of the total DIGITAL customer service system. We believe in training, not only for ourselves, but also for our customers. Your employees will, through educa-

tion, extract greater performance from your computer.

To provide this training, we have established completely equipped training centers in a dozen locations around the world. Our staff at these centers consists of full-time instructors dedicated to computer training.

The education DIGITAL offers includes standard and custom courses in both hardware and software. Our current schedule includes over 100 standard courses, ranging in duration from one to five weeks.

DIGITAL recognizes that you may have a unique requirement that can best be met with education at your location. Our on-site program meets this need by designing and conducting courses where and when you choose.

Types of Courses

Catalog Courses—Educational Services publishes its "Educational Courses Catalog" semi-annually. This catalog includes a full listing of hardware and software courses geared to DIGITAL's products, as well as many generic computer science courses. One section of this catalog is devoted to DECSYSTEM-20 training. Courses are listed with their schedules for the current six (6) month period. Prerequisites, course abstract, objectives and topic outlines are included.

Seminars—Conducted at our Regional Education Centers and at other convenient locations, seminars are designed to complement and expand upon our existing curriculum, as well as keep the customer informed of DIGITAL's product developments. By bringing these workshops to the customer's locale, Educational Services furnishes a convenient way for the DECSYSTEM-20 customer to gain additional information on his system.

On-Site Courses—If the customer has a group of individuals to train, it is often more economical to have Educational Services conduct courses at the customer's location rather than one of our Education Centers.

Custom Courses—If the customer has a unique application or training situation, custom courses are very effective. This approach gives the customer the added advantage of maximizing relevant material content while minimizing information extraneous to his operations.

DECSYSTEM 20 Training Program

Training Credits—Along with the purchase of a DECSYSTEM-20 the customer will receive ten (10) training credits. Training credits may be applied to the standard software courses for a period of one year from the date of issue.

If used as suggested below, the ten training credits will enable one system programmer, one system administrator, and one applications programmer to attend the appropriate software courses in Marlborough free of tuition charge:

System Programmer
DECSYSTEM-20 User (1 Credit)
DECSYSTEM-20 Assembly Language Programming (2 Credits)
DECSYSTEM-20 Systems Prog (1 Credit)
DECSYSTEM-20 Operation System (2 Credits)

System Administrator
DECSYSTEM-20 User (1 Credit)
DECSYSTEM-20 Systems Admin (1 Credit)

Applications
DECSYSTEM-20 User (1 Credit)

Programmer
DECSYSTEM-20 COBOL (1 Credit)

Supplementary Training—Each DECSYSTEM-20 customer's educational needs should be reviewed to determine if additional training (in excess of that provided by the ten training credits) is needed. If so, the additional requirements may be satisfied by the purchase of enrollments in courses conducted in Marlborough or on-site courses.

DECSYSTEM 20 Course Abstracts

DECSYSTEM-20 User
This course is intended for all users of the DECSYSTEM-20 wishing to gain knowledge of the structure of the system and its use from an interactive terminal viewpoint. This course serves as a prerequisite for all other courses in the curriculum.

DECSYSTEM-20 Assembly Language Programming
This course covers the DECSYSTEM-20 instruction set, user level I/O programming, and the MACRO assembler. This course is provided for the system programmer and serves as a prerequisite for the DECSYSTEM-20 Systems Programming Course.

DECSYSTEM-20 Systems Programming
This course covers the more advanced JSYS operations, interfacing with system programs,

and the structure of the operating system. This course is provided for the systems programmer and serves as a prerequisite to the Operating System course.

DECSYSTEM-20 Operating System

This course covers the internal working and algorithms of the operating system's Executive and Monitor. The course is intended for the senior systems programmer and should always be preceded by attendance in the Systems Programming and Assembly Language Programming course.

DECSYSTEM-20 Operator

This course is designed to provide the student with the knowledge required for the operation of a DECSYSTEM-20. This course must be preceded by the DECSYSTEM-20 User's Course.

DECSYSTEM-20 Systems Administration

This course is designed to provide the student with the necessary tools for administering a DECSYSTEM-20 installation. The course material is applicable to day-to-day operations, system controls, and available DIGITAL services.

DECSYSTEM-20 COBOL

This course is designed for the COBOL programmer desiring knowledge to write advanced COBOL programs on the DECSYSTEM-20.

DECSYSTEM-20 DBMS

This course is intended for initial users of DBMS including management, systems personnel and applications programmers.

Course Selection Guide

The following guide is provided to aid you in prescribing the right courses for the customer's various personnel.

Personnel Function	Suggested Course Sequence
User Operator	1. DECSYSTEM-20 User (1 Week) 2. DECSYSTEM-20 Operator (1 Week/ on-site)
Systems Programmer (Assembler)	1. DECSYSTEM-20 User (1 Week) 2. DECSYSTEM-20 Assembly Language Programming (2 Weeks) 3. DECSYSTEM-20 Systems Programming (1 Week) 4. DECSYSTEM-20 Operating System
Commercial Programmer (COBOL)	1. DECSYSTEM-20 User (1 Week) 2. DECSYSTEM-20 COBOL (1 Week) 3. DBMS (if applicable) (1 Week)
System Administrator	1. DECSYSTEM-20 User (1 Week) 2. DECSYSTEM-20 System Administration (1 Week)

Software Services

The DIGITAL software organization represents over 3,000 man-years of experience gained from the development and support of operating systems used in more than 50,000 computer installations. DIGITAL's software varies in complexity from real-time executives to minicomputers to the TOPS-10 and TOPS-20. Monitors of the DECsystem-10 and DECSYSTEM-20 applications cover the spectrum from process control and monitoring scientific experiments to implementing reservation and inventory control systems.

DIGITAL offers a wide range of software services. These services range from the personal attention of a skilled software consultant to the distribution of up-to-date software and software information. In this way you can get the most out of your DECSYSTEM-20 and keep pace with advancements in software.

You purchase only what you need. Consulting services are available on a short-term, per-call basis or for a longer-term scheduled or resident period. Likewise, software components, including manuals and updates, can be purchased as part of a software maintenance service or ordered separately from our Software Distribution Center.

Advanced Systems Group

The Advanced Systems Group, an integral part of the DECSYSTEM-20 product group, meets special customer's needs by augmenting the standard products and services of the DECSYSTEM-20. The Advanced Systems Group extends the range of the DECSYSTEM-20 product and services by providing:

Hardware/Software Design and Development—Engineering for special products and modifications to standard products, with emphasis in the areas of communications, network systems, high availability and redundant systems, and special configurations.

Systems Project Management—For large and complex configurations including such functions as systems evaluation and design, systems planning for future expansion, extended factory systems testing, special installation backup and support, and extensions to the DECSYSTEM-20 acceptance test procedures.

Repeat Manufacture—of previously designed and developed special products.

Advanced Systems Group products maintain DIGITAL's high standard of quality. All hardware products are supportable under DECSYSTEM-20 field service support plans and are supplied with full documentation, prints and diagnostics.

Support of special software is provided through a centralized support group on an individual basis for both installation and Software Performance Reports (SPR's).

Because of the customized nature of these products, the Advanced Systems Group provides individual system configuration review, system integration and system installation assistances when needed.

DECSYSTEM 20 Manuals

Documentation is an important part of the DECSYSTEM-20. The planning and production of manuals to support the user with clear, well-written descriptions of the system, programming languages, and command processor was an integral part of the development process.

Manuals are produced using on-line techniques. The text entry and editing capability of DIGITAL's interactive systems result in major user benefits. Because documentation files can be readily updated to reflect improvements made to the software, manuals can be more up-to-date.

Most software documentation is available in complementary quantities from your DIGITAL sales representative.

User Manuals

TITLE	CODE
ERROR DETECTION, RECOVERY REPORTING V1	EK-SEDRR-RF-001
BASIC USER'S GUIDE	DEC-20-LBMAA-A-D
COBOL REFERENCE MANUAL	DEC-20-LCRMA-A-D
COBOL UTILITIES MANUAL	DEC-20-LCUMA-A-D
FORTRAN REFERENCE MANUAL	DEC-20-LFRMA-A-D
MACRO ASSEMBLER REF. MANUAL V50	DEC-20-LMRMA-A-D
BATCH OPERATOR'S GUIDE	DEC-20-OBOGA-A-D
VA-D BATCH REFERENCE MANUAL	DEC-20-OBRMA-A-M
COMMAND PARSER SPECIFICATION	DEC-20-OCPSA-A-D
LINK REFERENCE MANUAL	DEC-20-ULRMA-A-D
MONITOR CALLS REF. MANUAL	*DEC-20-OMRMA-A-D
MONITOR CALLS USER'S GUIDE	DEC-20-OMUGA-A-D
DEC-20 SORT OPERATOR'S GUIDE	DEC-20-USTGA-A-D
DEC-20 USER'S GUIDE	**DEC-20-OTPGA-A-D
ACCOUNTING SYSTEM SPECIFICATIONS	***DEC-20-OUGAA-A-D
GETTING STARTED WITH RUNOFF VA	DEC-20-UASSA-A-D
GETTING STARTED WITH DECSYSTEM-20	DEC-20-URGSA-A-D
SOFTWARE INSTALLATION GUIDE	DEC-20-XGSAA-A-D
DATA BASE MANAGEMENT SYSTEM PROGRAMMER'S PROCEDURES MANUAL	DEC-20-XSIGA-B-D
SYSTEM MANAGER'S GUIDE	DEC-20-APPMB-A-D
	DEC-20-OSMGA-B-D

*plus update DEC-20-OMRMA-A-D-DN1
**plus update DEC-20-OTPGA-A-D-DN1
***plus update DEC-20-OUGAA-A-D-DN1

TERMINAL SESSION

Logging In

To begin a DECSYSTEM-20 session, the user types CONTROL-C (^C). If available, TOPS-20 responds with the system identification, date and version of the command processor.[1] The user is then prompted with an @ and begins the LOGIN process. [In these examples, the user has used the command recognition and prompting ("novice") modes; TOPS-20 allows the user to type only as much of any command as is necessary for uniqueness.] He then types the escape character, and the command processor determines what command it is and prompts for the next argument. In our example, the colored portions indicate the user's typed input, while the black typeout is produced by the system.

The user types LOG followed by escape and is prompted for a USER.[2] Being a bit uncertain about this, he types a ? and the system defines the field as USER NAME[3] and reprompts up to the last point.[4] The user name DEMO-1 is entered followed by an escape, and the TOPS-20 exec prompts for a password. The password is not echoed to the terminal for security, and the user is then prompted for an account number. The line is terminated by a carriage return. If the login is successful, the user is assigned a Job Number[5] and then prompted for input. The two @ signs indicate the return to command level from a user level command.[6]

```
1 V 1.02.37, 1031 Development Sys., 11-MAR-76. TOPS-20 1A(103)
2 @LOGIN (USER) ?
3 USER NAME
4 @LOGIN (USER) DEMO-1 (PASSWORD)  (ACCOUNT #) 10010
5  JOB 16 ON TTY2 12-MAR-76 14:38
6 @
```

Building a FORTRAN Program

Now the user is ready to begin work. He specifies that he wants to edit a FORTRAN source file called TEST.FOR.[7] Since the file does not yet exist, the editor warns him that a new file is being created,[8] and prompts with line number 00100.[9] The program is now entered line-by-line; each time the editor prompts with a new line number. Input is terminated with an escape (the editor prints it as a $[10] and the user enters command level in the editor, indicated by the asterisk.[11] The programmer notices that he hasn't given A an initial value, and inserts it as line 50. The E command then ends the editing process[12] and the file name is printed as the file is closed.[13]

```
7 @EDIT TEST.FOR

8 %File not found, Creating New file
  Input: TEST.FOR.1
9 00100             DO 5 X=1,1000
  00200             DO 5 J=1,100
  00300    5        A=( (X-1)/(X+1) )%.39*X + A
  00400             WRITE(5,100) A
  00500    100      FORMAT(' A IS ' , E13.6)
  00600             END
10 00700    $
11 *I50
  00050             A=0.
12 *E

13 [TEST.FOR.1]
```

Executing the Program

The user now attempts to execute the program by an EX command (again using prompting) and specifying the file name.[14] TOPS-20 determines from the file extension FOR that this is a FORTRAN program, and calls the compiler automatically.[15] The compiler detects a fatal error in line 300[16] and the LINKing phase, invoked automatically by the EXECUTE command, is bypassed.[17] The user asks for the editor again, and the system assumes, since it wasn't specified otherwise, that he wants to work on the same file he was using the last time he used the editor—TEST.FOR.[18] The user has line 300 printed for reference[19] and then instructs the editor to search for a % and replace it by an * in line 300.[20] The editor retypes the corrected line. The user now leaves the editor with a G command,[21] which causes the new version of the program TEST.FOR.2[22] to be closed (TEST.FOR.1 is still preserved) and specified that the last execution command is to be retrieved (back in).[14]

```
14 @EXECUTE (FROM) TEST.FOR
15 FORTRAN: TEST
16 00300    5        A=( (X-1)/(X+1) )?.39*X + A
   ?FTNIAC LINE:00300 ILLEGAL ASCII CHARACTER % IN SOURCE

   ?FTNFTL    MAIN.          1 FATAL ERRORS AND NO WARNINGS
17 LINK:    Loading
   [LNKNSA No start address]

   EXIT
   @
   @EDIT
18 Edit: TEST.FOR.1
19 *P 300
   00300    5        A=( (X-1)/(X+1) )%.39*X + A
   *
20 *S%$*$300
   00300    5        A=( (X-1)/(X+1) )*.39*X + A
21 *G

22 [TEST.FOR.2]
```

Retry the Program

The FORTRAN compiler is called automatically[23] and this time successfully compiles TEST. The linking loader LINK loads the program and starts execution.[24] Line 400 of the program causes A to be displayed[25] and the program terminates successfully.[26] The user is returned to command level, at which time he inquires (note the recognition and prompting) about his job.[27] This process can be abbreviated by a user once he becomes familiar with the commands.[28] The user also asks about his quotas of disk storage space,[29] and the system tells him he is allowed 3,000 working pages (while logged in) and 2,000 pages of permanent storage (when logged out). The system has a total of almost 28,500 pages available for use.

```
23 FORTRAN: TEST
   MAIN.
24 LINK:    Loading
   [LNKXCT TEST Execution]
25 A IS  0.194420E+08

   END OF EXECUTION
26 CPU TIME: 2.02  ELAPSED TIME: 2.89
   EXIT
   @
27 @INFORMATION (ABOUT) JOB-STATUS
    JOB 16, USER DEMO-1, ACCT 10010, TTY2
28 @I J
    JOB 16, USER DEMO-1, ACCT 10010, TTY2
   @
29 @INFORMATION (ABOUT) DISK-USAGE (OF DIRECTORY)
    6(7) PAGES ASSIGNED, 3 IN USE, 3 DELETED
    3000 WORKING, 2000 PERMANENT ALLOWED
    28500 SYSTEM PAGES FREE
```

Debugging the Program

Although this is a simple example, the user asks to use the interactive FORTRAN debugger to examine the execution of his program.[30] The DEBUG command causes this to happen by recompiling the program, loading it, and starting the debugger,[31] which prompts with a >>. The user now realizes that he needs a clean copy of the program to refer to, and suspends the debugger with a CONTROL-C.[32] He issues a PUSH,[33] which provides him with a separate process (identified by the command processor version number) and then types out his program with a TYPE command.[34] Note that he is not disturbing the suspended debugging session by doing this. The POP[35] command releases this new process and the CONTinue returns to the original task.[36] The user now sets a breakpoint, or pause, at line 300[37] and starts the program running.[38] Before executing that particular line, the FORDDT system regains control and allows the user to type out value of variables in the program. X is typed as a real, J as an integer.[39] When the program is commanded to continue, it stops at line 300 (statement number 5) and waits again.[40] The single command "T"[41] says "type what I last asked for" and we now note that J has changed value. A single "C" causes a continue,[42] and this time the user displays X and A.[43] To change the value for A, he issues an ACCEPT, specifies floating point format, and enters a −1000.[44] Again, a simple continue[45] and the program pauses once more. This time just A is displayed (its value is the new one he entered).[46] He now removes the pause,[47] and continues.[48] The program now runs to completion and exits. Note that the elapsed time for this debugging process was only one minute and thirty-three seconds.

```
   @
30 @DEBUG (FROM) TEST/COMPILE/DEBUG
   FORTRAN: TEST
   MAIN.
   LINK:    Loading
   [LNKDEB FORDDT Execution]

31 STARTING FORTRAN DDT

32 >> ^C
33 @PUSH
    TOPS-20 1A(103)
34 @TYPE TEST.FOR
   00050              A=0.
   00100              DO 5 X=1,1000
   00200              DO 5 J=1,100
   00300    5         A=( (X-1)/(X+1) )*.39*X + A
   00400              WRITE(5,100) A
   00500    100       FORMAT(' A IS ' , E13.6)
   00600              END
   @
35 @POP
36 @CONT


37 >> PAUSE #300

38 >> START

   PAUSE AT 5 IN MAIN PROGRAM

39 >> TYPE X,J/I

   X         =     1.000000

   J         =   1

40 >> CONTINUE


   PAUSE AT 5 IN MAIN PROGRAM

41 >> T

   X         =     1.000000

   J         =   2

42 >> C
```

```
   PAUSE AT 5 IN MAIN PROGRAM

43 >> T X,A

   X          =     1.000000

   A          =    0.0000000

44 >> ACCEPT A/F -1000.

   A          =    -1000.000

45 >> C


   PAUSE AT 5 IN MAIN PROGRAM

   >> T A

   A          =    -1000.000

47 >> REMOVE

48 >> CON

   A IS  0.194410E+08

   END OF EXECUTION
   CPU TIME: 2.48  ELAPSED TIME: 1:36.30
   EXIT
```

Optimization of the Program

Believing his program to be correct, the user now executes it again, this time asking FORTRAN to optimize the code for faster execution.[49] Comparing the CPU time here to that before,[26] we note the program now runs over three times faster. As a test, the user now wants to direct the terminal output from his WRITE statement to a disk file. To do so, he simply DEFINE's (asking for help with the "?"),[50] the logical device 5 to be on disk (DSK:).[51] Now when the program is executed, we do not see the output on the terminal.[52]

```
49 @EXECUTE TEST/COMPILE/OPT
   FORTRAN: TEST
   MAIN.
   LINK:    Loading
   [LNKXCT TEST Execution]
   A IS  0.194420E+08

   END OF EXECUTION
   CPU TIME: 0.38  ELAPSED TIME: 0.55
   EXIT
   @
50 @DEFINE (LOGICAL NAME) ?
    LOGICAL NAME TO DEFINE OR DELETE,
    OR "*" TO DELETE ALL
```

```
51 @DEFINE (LOGICAL NAME) 5 (AS) DSK:
   @
52 @EX
   FORTRAN: TEST
   MAIN.
   LINK:    Loading
   [LNKXCT TEST Execution]

   END OF EXECUTION
   CPU TIME: 0.45  ELAPSED TIME: 1.13
   EXIT
   @
```

Getting a Directory and Printing a File

To find the file, the user now asks for a directory of his disk area. Terminating the line with a comma[53] puts him in subcommand mode, which is indicated by the double @ @. He now specifies reverse order[54] sort chronologically[55] by creation date.[56] A blank subcommand line terminates that mode and the directory is typed. The newest file is FOR05.DAT.3, the result of a FORTRAN program writing on device 5.[57] The user interrupts the list by CONTROL-C,[58] and asks for the data file to be typed.[59] To get a listing of the file on the line printer, the user issues a LIST command.[60] To see how it will be before the file is printed, he issues a QUEUE inquiry,[61] which shows that his job is being printed now on physical line printer 0. The 24 pages is a maximum limit to prevent accidentally printing excessive worthless pages.

```
53 @DIRECTORY (OF FILES) ,
54 @@REVERSE (SORTING)
55 @@CHRONOLOGICAL (BY) ? ONE OF THE FOLLOWING:
    CREATION
    READ
    WRITE

56 @@CHRONOLOGICAL (BY) CREATION
   @@

      <DEMO-1>
57  FOR05.DAT.1
    TEST.REL.5
      .FOR.2
58    .QOR.1^C
   @
59 @TYPE FOR05.DAT.1
    A IS  0.194420E+08
   @
   @I JOB
    JOB 16, USER DEMO-1, ACCT 10010, TTY2
   @
60 @LIST (FILE) ? FILE NAME
   @LIST (FILE) TEST.FOR
61 @QUEUE
```

```
OUTPUT QUEUE:
DEV            USER                    JOB     SEQ PRIO LIMIT    AFTER
PLPTO * AARON                          T1003D  441  10      36
LPT     DEMO-1                         TEST    442  10      24
* Job beins output now

TOTALS: LPT:    2 Jobs;   60 Pages
```

Finding Out What's Going On

Finally, the user asks for a SYSTAT.[62] Among the information printed is a list of all current users by job, what line they are on, what program they are running (EXEC is the command processor) and their names.[63] Operator service jobs are listed last. In this example, the system has been up for three and a half hours since timesharing began and 26 jobs are in use. Over the last three scheduling periods, the load of jobs in the processor run queue has been 3.8, 2.8, 2.9.

Satisfied with his session, the user logs out,[64] having used 18 seconds of compute time in over 14 minutes of elapsed time.

```
@
62 @SYSTAT
   FRI 12-MAR-76 14:52:24   UP 5:43:01
   16+5 JOBS    LOAD AV    3.8     2.8     2.9

   SYSTEM IS TIMESHARING

63  JOB   LINE PROGRAM   USER
      2     23  EXEC      MCKIE
      7     15  RUNOFF    KIRSCHEN
      8     63  EDIT      9STUDENT
      9     14  EXEC      MILLER
     10     65  EXEC      3STUDENT
     11     11  EXEC      PORCHER
     12     13  EXEC      HALL
     13      3  EXEC      AARON
     14     26  EXEC      EXERCISER
     15     22  EXEC      HURLEY
     16*     2  EXEC      DEMO-1
     17      4  TECO      FRIES
     19     70  EDIT      11STUDENT
     20     71  EDIT      8STUDENT
     21     62  EXEC      10STUDENT
     23     43  IPRNEW    HEMINGWAY

      1    101  PTYCON    OPERATOR
      3    106  EXEC      OPERATOR
      4    105  OPLEAS    OPERATOR
      5    103  LPTSPL    OPERATOR
      6    104  BATCON    OPERATOR
64 @LOGOUT
  KILLED JOB 16, USER DEMO-1, ACCT 10010, TTY 2, AT 12-MAR-76 14:53:05
    USED 0:0:18 IN 0:14:37
```

☐ Please have a sales representative call me.

☐ Please send me additional information on the DECSYSTEM-20.

NAME _____

COMPANY/INSTITUTION _____

_____

STREET _____

CITY _____

COUNTRY _____

TELEPHONE _____

My application is _____

_____

_____

_____

_____

_____

_____

_____

_____

☐ Please have a sales representative call me.

☐ Please send me additional information on the DECSYSTEM-20.

NAME _____

COMPANY/INSTITUTION _____

_____

STREET _____

CITY _____

STATE _____ ZIP _____

TELEPHONE _____

My application is _____

_____

_____

_____

_____

_____

_____

_____

_____

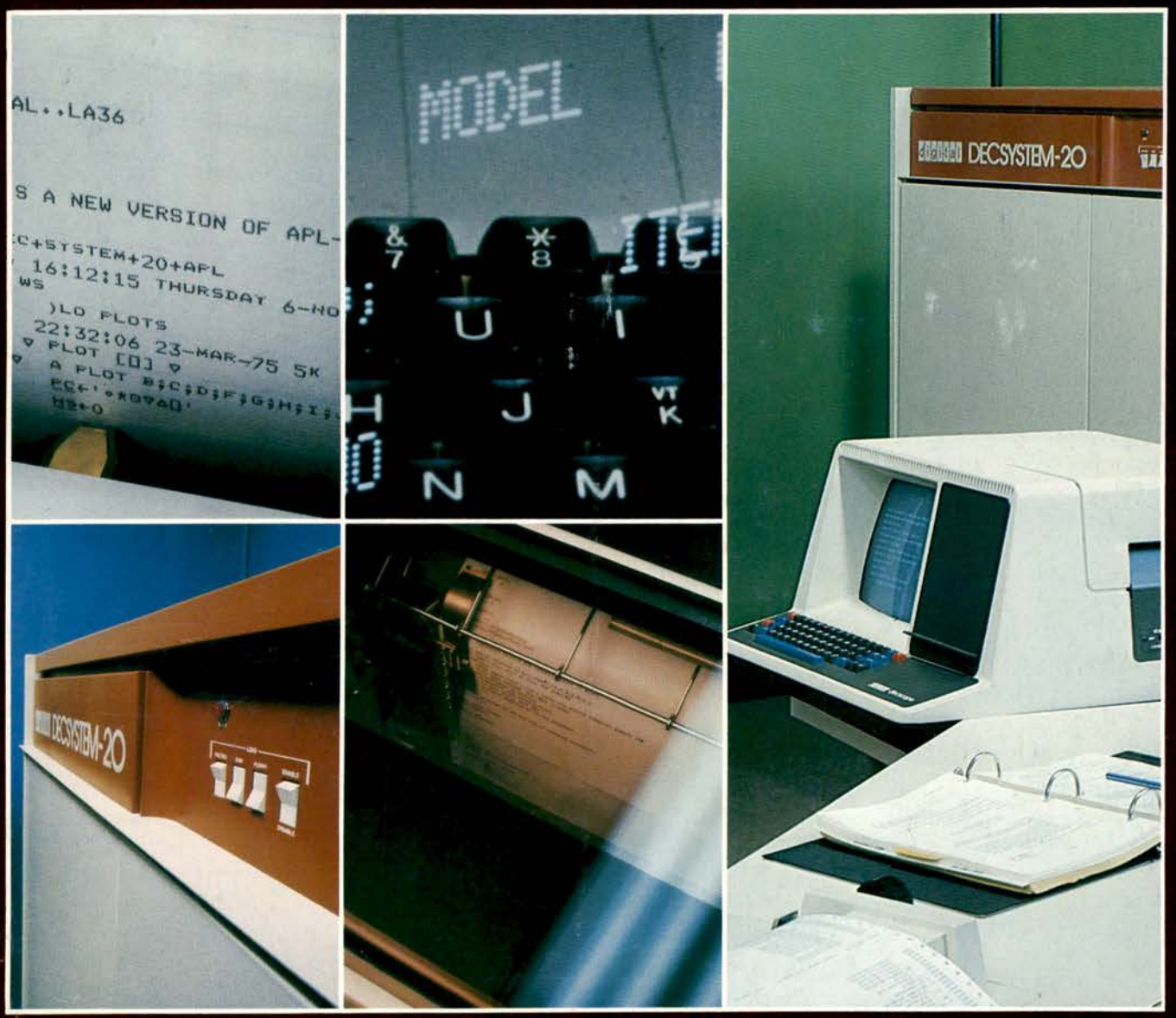
DECSYSTEM20 Technical Summary

digital

DIGITAL EQUIPMENT CORPORATION
Large Computer Group
Marlborough, Massachusetts 01752
In Europe: Digital Equipment Corporation International, Geneve 26, Switzerland

digital

DECSYSTEM20 Technical Summary



The best of the big systems with the best of the small

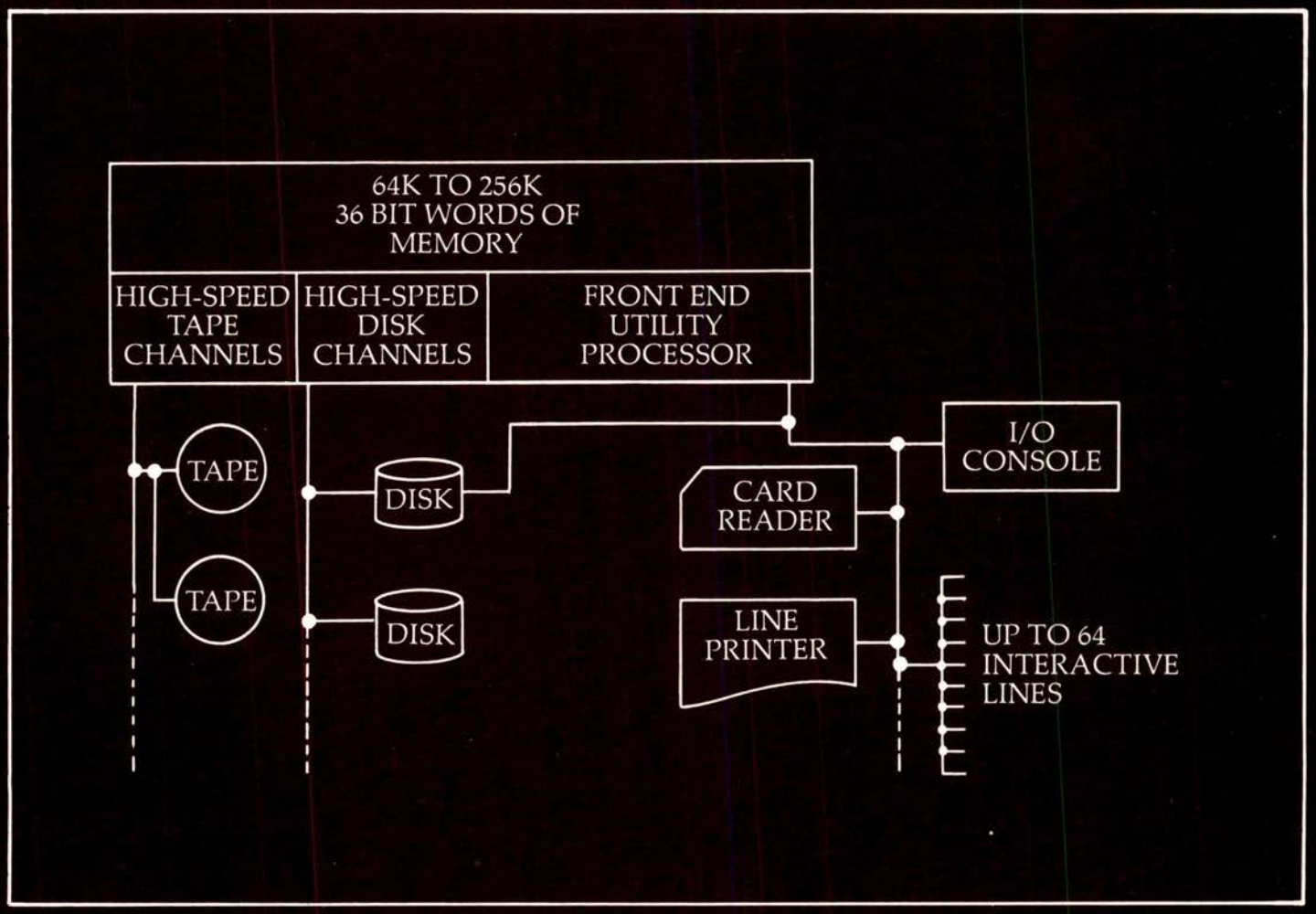


Table of Contents

DECSYSTEM 20 Introduction

Digital Equipment Corporation is pleased to present the DECSYSTEM-20—a medium-scale computer with large computer capabilities available at small computer prices.

The DECSYSTEM-20 features a high-performance, virtual memory system that provides a multi-tasking, multi-programming environment to support concurrent time-sharing, batch and transaction processing.

The DECSYSTEM-20 is easy to use and easy to maintain. It's flexible, expandable, reliable, efficient and above all—affordable.

The DECSYSTEM-20 is the result of over ten years experience in the design and development of large scale multi-language timesharing systems. It was in 1964 that DIGITAL installed the first commercially available interactive timesharing system. And it was no accident.

Even then, DIGITAL wanted people to make better use of their time to be more productive, to get their "hands on" the computer.

Building on the success of these early systems, DIGITAL introduced the PDP-10 in 1967. It featured improved technology and better peripherals. The operating system had a better human interface so it was easier to use. Because of its success, development activity accelerated and in 1971, DIGITAL introduced the DECsystem-10 family of computing systems with a multi-mode operation system, called TOPS-10, offering time-sharing, batch processing, real-time and communications capability to support computer based remote stations.

Over 450 DECsystem-10 systems are now installed in 21 countries worldwide.

The design, manufacture and support of these large systems provided DIGITAL with the advanced technology experience necessary for the development of the DECSYSTEM-20.

DIGITAL also put its mini-computer expertise to work in the design of the DECSYSTEM-20. The central processor, memory, and I/O controllers are integrated into one neat, compact unit requiring a minimum amount of floor space—only twenty-eight square feet for processor, memory and I/O controllers. This cost-saving packaging is made possible because the DECSYSTEM-20 uses the latest in state-of-the-art technology. The processor is built of high-density, multi-layer circuit boards and emitter-coupled logic (ECL). Not only is the DECSYSTEM-20 smaller, but it's faster and more reliable.

The DECSYSTEM-20 features a large and powerful instruction set. The programs that the user writes will take advantage of this feature because the compilers and interpreters produce less machine code than do comparable systems. And because programs are smaller, they will run faster. The instruction set, despite its size, is easy to learn because it's logically grouped into "families" of instructions. One such family is the Business Instruction Set. It has sophisticated editing and character handling capabilities for program development or day-to-day problem solving. And the entire instruction set is micro-programmed because that's the cost-effective way to implement all these capabilities.

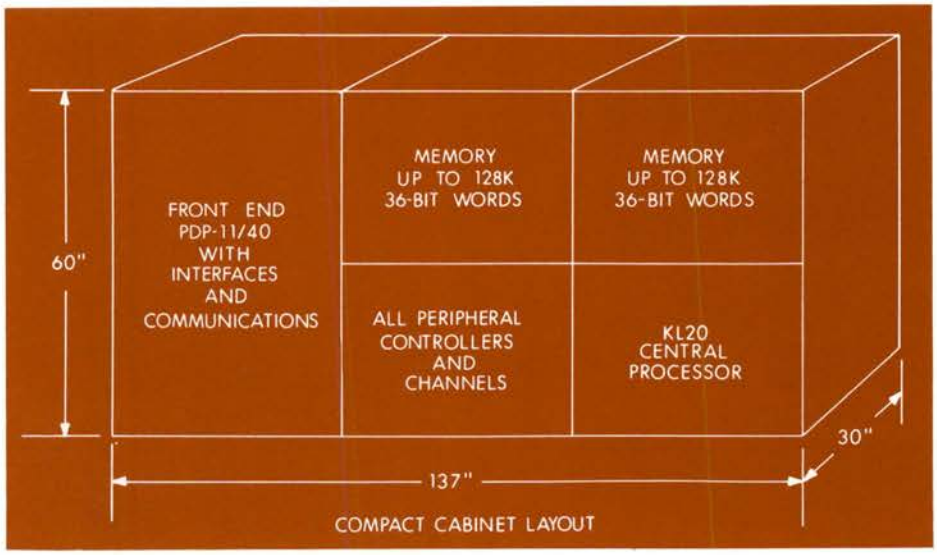
The DECSYSTEM-20 supports up to one million bytes of high-speed, low-cost memory. To ensure the reliability of operation and integrity of data, all memory read and write operations are parity checked.

The system features integrated high-speed data channels and mass storage controllers for disk and magnetic tape. These integrated I/O controllers permit the simultaneous transfer of data and control information. Each controller has its own direct path to memory; hence I/O operations on two separate controllers can be overlapped for greater throughput and efficiency. Since each controller may access up to sixteen words of data in a single request to or from memory, the DECSYSTEM-20 has an I/O bandwidth of more than 7 million bytes per second.

The central processor contains accounting and analysis hardware in the form of clocks and timers to provide accuracy and efficiency in the control and operation of the DECSYSTEM-20.

A PDP-11 based front end processor enables the central processor to do more work by handling unit-record peripherals, console operations, and terminal communications. Equally important is its role as diagnostic computer for the central processor. Even if the main system is completely inoperative, maintenance engineers can utilize the front end computer for module level diagnosis. Preventative maintenance may be done during timesharing operations. Both features increase system performance by reducing Mean Time to Repair and stand-alone preventative maintenance time.

The virtual memory hardware was designed in conjunction with the new TOPS-20 operating system to provide the most efficient and reliable system possible. As a consequence, programs that are larger than available core may be efficiently run without overlays. The paging hardware maps core for both the operating system and user programs, and provides information needed by the software to efficiently allocate core. Because the monitor is modular, demand paged, and write-protected, it is more reliable. The resident portion is quite small, thereby increasing system throughput by making more core available for user programs. User core is dynamically allocated as needed. No core is wasted since there are no partitions.



COMPACT CABINET LAYOUT

DECSYSTEM 20 Features Simplicity

Users may access the system by timesharing or through batch—whichever mode is the most appropriate to their needs. Production jobs may be entered into batch by cards or from a terminal. The same command and device specifications are used under both timesharing and batch. The user needs to learn one simple command language for both modes. Batch efficiency is high because the system can process multiple batch jobs at one time.

All user programs may communicate with each other in a variety of ways. Programs may be organized in a hierarchical control structure which provides greater reliability and security for user and system software. The failure of one program will not affect the others.

Data in the DECSYSTEM-20 file system is secure and reliable —by design. These features are especially significant because data and program sharing in memory is also a key feature. This controlled yet flexible data sharing facilitates powerful data base management systems.

Device independence and extremely flexible I/O access methods make programming easier and more efficient. Program execution is also enhanced since the I/O devices may be chosen at run time. Other features which increase system throughput include simultaneous update facilities and spooling of unit record devices such as the line printer and card reader.

Timesharing increases productivity by allowing the user to create programs using the on-line editor, and then immediately, from the same terminal, to compile and test them using on-line debugging techniques. Program development time is greatly reduced since many tests may be run in one session—no waiting for hours to see if there were errors. Data preparation is also much faster and easier since data files may be created on-line. Memos may be easily sent to all users via a mailbox facility, and documents may be formatted and printed using the document preparation program.

The command language provides easy access to these facilities. It prompts the novice user by guide words, explanations (if desired) and clear English error messages. At the same time, the expert user can enter commands quickly in an abbreviated form which will be recognized and acted upon by the command language processor. Thus, both novice and expert become more efficient and productive.

All DECSYSTEM-20 languages enhance program production. They contain source level debuggers, may be used under both batch and timesharing, and produce sharable, optimized object code. COBOL, FORTRAN, BASIC, APL, ALGOL, CPL, MACRO—the programmer may choose whichever is best suited to his application and expertise. ANSI-68 COBOL supports ISAM, high performance sorting, simultaneous updating, and DBMS—Digital's data base management system. COBOL utilizes the business instruction set for greater object code efficiency. FORTRAN, ALGOL and APL provide a variety of solutions to scientific problems, while BASIC, APL and CPL are designed and implemented to aid the beginning programmer.

The system administrator has full control over system access and usage and can determine who may use the system, what level of privileges they may have, and how the resources are allocated. Accounting tools enable him to accurately charge users for the resources they have actually used.

System operation is extremely simple. There is no monitor generation required. Operator procedures are very simple and the operator privileges are restricted to those necessary for normal system operation, enhancing system security and reliability. Flexible backup facilities enable single files to be saved and restored as well as the entire system.

The remote on-line diagnostic system enables maintenance engineers to do preventative maintenance during timesharing and to diagnose the system quickly and easily—thereby reducing Mean Time to Repair and enhancing system availability.

DECSYSTEM20 Hardware

System Architecture

The system architecture of the DECSYSTEM-20 is the result of design goals which included small size, high throughput, and high reliability and maintainability. System logic, housed in low profile cabinets, is made up of high density ECL circuitry on multi-layer boards. The Central Processor, memory, and mass storage controllers are integrated into a single cabinet assembly, requiring reduced floor space and cabling. Within the same assembly, a PDP-11 front end processor handles unit record devices, asynchronous communications, and system diagnostics. Mass storage devices are linked to the controllers over high-speed Massbus data and signal paths. Separate Massbusses provide increased throughput for tapes and disks. Unit record devices link directly to controllers attached to the PDP-11 UNIBUS. The overall design was heavily influenced by DIGITAL's advanced manufacturing technologies, developed and refined both in the mass production of minicomputers and in the production of the large-scale DECsystem-10. The resulting manufacturing techniques provide customers with a powerful hardware system and fully developed large computer software at low cost.

Central Processor Architecture

The Central Processor uses the word length and instruction set of the latest processor developed for the DECsystem-10 computer line.

The Instruction Execution Unit performs instruction execution, arithmetic operations, program-controlled I/O, and all clocking and metering functions performed by the DECSYSTEM-20. Included within the unit are eight sets of sixteen high-speed registers, a 1280 x 75 bit microstore for instruction logic, a 512 x 16 bit high-speed dispatch RAM, a seven level priority interrrupt system with vectoring, plus system clocks and performance meters.

The Memory Control Unit provides paths and logic for all transfers into and out of the DECSYSTEM-20 memory. Virtual-to-physical address translation is performed by a 64 x 154 bit RAM. The Memory Control also includes a series of multiple word channel buffers for high-speed direct I/O over the Channel Bus to the individual Massbus devices. For each device type, there is one channel containing a 15-word data buffer, a channel command word register, and a control word location pointer—effectively a program counter. The channels perform data transfer by executing channel programs which are set up in memory by device service routines.

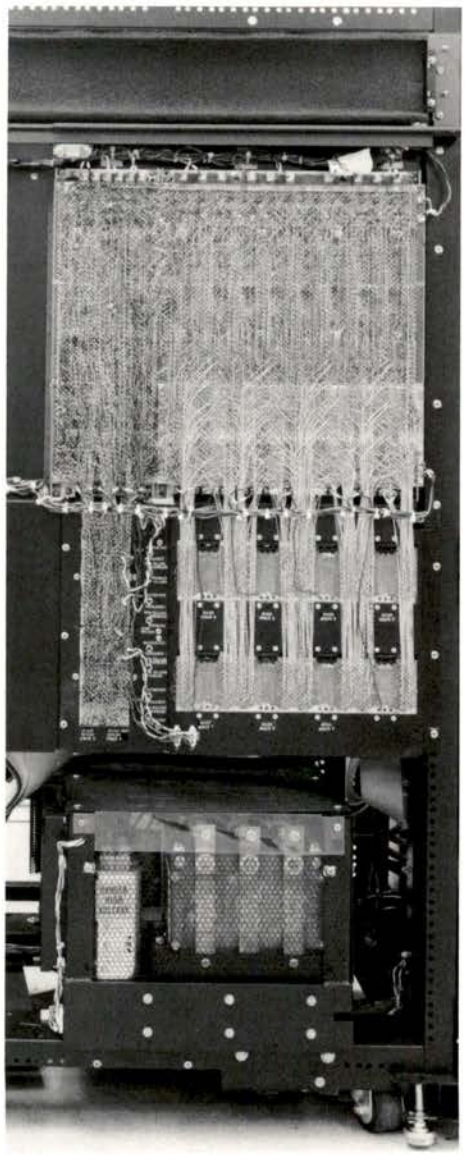
A Storage Bus provides a 1.6 million words/second transfer bandwidth between the Memory Control Unit and the DECSYSTEM-20 memory. In diagnostic mode, the Storage Bus is used to access the internal memory controllers. The Channel Bus is a physically short, high-speed data transfer path providing a peak I/O bandwidth of six million words/second between the Memory Control Unit and the Massbus controllers. The Channel Bus operates in a synchronous time division multiplexed mode, allowing the connection of multiple controllers to memory.

An Execution Bus connects the front end interface to the Instruction Execution Unit. The bus operates asynchronously with a bandwidth up to .5 million words/second for I/O instruction transfer.

A high-speed data path (Massbus) connects the integrated channel controllers and mass storage devices, such as disk or tape. Operating either asynchronously or synchronously, the Massbus transfers control and status information or blocks of data between devices and controllers. The UNIBUS, an integral part of the PDP-11 architecture, provides a data and status control path between the Front End Processor and the devices making up the PDP-11 system, including the communications interface and unit record I/O devices.

Dual access disks link directly into the PDP-11 disk controller for diagnostic and system initialization procedures. Integrated channel Massbus controllers allow simultaneous execution of non-data commands, such as seek, rewind, etc., while another device, linked to the same controller, is transferring data. To improve disk transfer rate, Massbus Controllers include a double buffer arrangement so that the software can pre-load the next transfer request during the current transfer. Using this capability, the system is able to read data corresponding to two separate requests from sequential sectors on the same disk without incurring rotational delay. Upon the completion of a channel operation, the controller generates a priority interrupt to the Central Processor. When the interrupt is recognized, control is passed directly to the appropriate routine through an interrupt vector supplied by the controller.

The Front End Interface provides for high-speed simultaneous two-way transfers of variable-byte data between the PDP-11 processor and the DECSYSTEM-20 Central Processor. The unit permits two-way doorbell-type interrupts between the processors and provides the access path to the Execution Bus for all diagnostic and bootstrapping functions.



Central Processor Features

The Central Processor directs the operation of the entire DECSYSTEM-20. It contains a microcoded instruction set, eight sets of sixteen general purpose registers, five accounting and performance meters, and instruction interrupt and trap facilities.

The Instruction Set is implemented in microcode, allowing for easy maintainability and expansion. The Instruction Set is a superset of that of the DECsystem-10 family of computers and includes 385 logically grouped instructions. A number of operation codes are also included with the Instruction Set which trap either to the TOPS-20 monitor or to the user's area, allowing a full range of programmable monitor calls. Although the number of instructions is large, they are easy to learn because of a logical grouping into families with consistent mnemonic code format. All instructions are capable of directly addressing a full 256K words of memory without resorting to base registers, displacement addressing, or indirect addressing. Instructions may, however, use indirect addressing and indexing to any level. Most instruction classes, including floating-point, allow immediate mode addressing, where the result of the effective address calculation is used directly as an operand in order to save storage and execution time.

Instruction Set

Half-word Data Transmission—These instructions move a half-word, optionally modifying the contents of the other half of the destination location. These 64 instructions differ in the direction they move half-words and in the manner they modify the other half of the destination location contents.

Full-word Data Transmission—The full-word data transmission instructions move one or more full words of data. The instructions may perform minor arithmetic operations, forming the negative or the magnitude of the word being processed.

Byte Manipulation—Four byte manipulation instructions pack or unpack bytes of arbitrary length anywhere within a word. Two instructions are provided for updating byte pointers.

Logic Instructions—The logic instructions provide the capabilities of shifting and rotating, as well as performing the complete set of 16 Boolean functions of two variables.

Floating-point Arithmetic—These instructions perform scaling, negating, addition, subtraction, multiplication, and division in single and double precision, floating-point format. In the single-precision floating-point format, one bit is reserved for the sign, eight bits are used for the exponent, and 27 bits are used for the fraction. In double-precision floating-point format, the fraction is extended to 62 bits in the second word. The sign bit of the second word is unused.

Fixed/Floating Conversions—Special instructions provide the capability for converting fixed-point numbers to or from floating-point numbers. Two sets of instructions are provided to perform this function: one set meeting the FORTRAN standard, a second set meeting the ALGOL standard.

Arithmetic Testing—The arithmetic testing instructions jump or skip depending on the result of an arithmetic test, and optionally perform an arithmetic operation on the test word.

Logical Testing, Modification, and Skip—These instructions modify, test, and/or skip using a mask of selected bits in an accumulator.

Program Control—This class includes conditional and unconditional jump instructions. In addition, it contains powerful instructions for efficient calls for short subroutines, calls for subroutines with many parameters and stack-oriented subroutine calls.

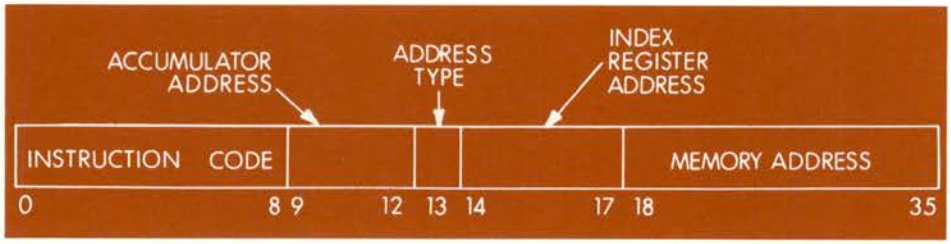
Input/Output Operations—Input/Output instructions govern direct transfers of control information to and from the peripheral equipment controllers and also perform certain operations within the processor. Block transfer instructions can handle multiple word data transfers to and from direct I/O devices.

Business Instruction Set—Five instructions comprise the Business Instruction Set in the central processor. Four are arithmetic instructions to add, subtract, multiply, and divide using double-precision, fixed-point operands. The fifth is a STRING instruction capable of performing nine separate functions, including:

- An EDIT Capability
- Decimal-to-binary and binary-to-decimal conversion in both offset and translated mode
- Move STRING in both offset and translated mode
- Compare STRING in both offset and translated mode (Offset Mode is byte modification by addition of the effective address of the STRING instruction to the source byte. Translated Mode is byte modification by translation through a table of half words located at the effective address of the STRING instruction. This also occurs in EDIT. In addition to providing the translation function, those instructions which use translation can control the flags in AC's and can detect special characters in the source.)

The Extended Business Instruction Set provides for faster processing of COBOL programs since there are specific instructions for doing more comprehensive string operations which commonly arise in such applications.

Instruction Format



The nine high-order bits (0-8) specify the operation; bits 9-12 usually address an accumulator but are sometimes used for special control purposes, such as addressing flags. The rest of the instruction word supplies information for calculating the effective address, which is used for immediate mode data or is the actual address used to fetch the operand or alter program flow. Bit 13 specifies the type of addressing (direct or indirect); bits 14-17 specify an index register for use in address modification (zero indicates no indexing); and the remaining eighteen bits (18-35) contain the address. The codes not implemented as instructions trap and are interpreted by routines included for the purpose.

Number System—Arithmetic instructions in the DECSYSTEM-20 use two's complement, fixed-point conventions to do binary arithmetic. In a word used as a number, bit 0 (the leftmost bit) represents the sign, 0 for positive, 1 for negative. In a positive number, the remaining 35 bits represent the magnitude in ordinary binary notation. The negative of a number is obtained by taking its two's complement. Zero is represented only by a word containing all 0's.

Fixed-point Arithmetic. Two conventions define a number as an integer (binary point at the right) or as a proper fraction (binary point at the left). In these two cases, the range of numbers represented by a single word is $-(2)^{35}$ to $(2)^{35}-1$ or -1 to $1-(2)^{-35}$. Since multiplication and division make use of double-length numbers, there are special instructions for performing these operations to yield results that can be represented by a single word for single precision operations.

The format for double-length fixed-point numbers is an extension of the single-length format. The magnitude (or its two's complement) is the 70-bit string in bits 1-35 of the high- and low-order words. Bit 0 of the high-order word is the sign, and bit 0 of the low-order word is ignored. The range for double-length integer and proper fractions is thus $-(2^{70})$ to $2^{70}-1$ or -1 to $1-(2^{-70})$.

Floating-point Arithmetic. A double-precision word consists of the sign, an 8-bit exponent and a 62-bit fraction. This gives a precision in the fraction of 1 part in 4.6×10^{18} and an exponent of 10 to a power of from -38 to $+38$.

Processor Modes—Instructions are executed in one of two modes. Programs operate in either User Mode or Exec Mode. In Exec Mode operation, all implemented instructions are legal. The monitor operates in Exec Mode and is able to control all system resources and the state of the processor. In User Mode operation, certain instructions are illegal, causing monitor traps when executed. Functions such as User Mode I/O are done through calls to the monitor, issued by the user program.

Executive and User Modes are further divided into two submodes each.

Processor Modes

User Mode

Public Submode	Concealed Submode
• User Programs • 256K Word Address	• Proprietary Programs • Can READ, WRITE, EXECUTE or TRANSFER to any location labeled Public
• All instructions permitted unless they compromise integrity of system or other users • Can transfer to concealed submode only at entry points	

Executive Mode

(Monitor)

Supervisor Submode	Kernel Submode
• Performs general management of system • Performs those functions that affect only one user at a time • Executes in virtual address space labeled Public	• Performs I/O for system • Performs those functions that affect all users

Fast Register Blocks—General purpose registers are another DECSYSTEM-20 feature that help improve program execution. Eight sets of sixteen fast integrated circuit registers can be used as accumulators, index registers, and as the first sixteen locations in memory. Since the registers can be addressed as memory locations, they do not require special handling. How the registers are used depends on how they are addressed in the instruction word.

Program switching time between register sets is 500 nanoseconds.

Different register blocks can be used by the operating system and individual users. This eliminates the need for storing register contents when switching from User Mode to Exec Mode.

Accounting and Performance Meters—Five accounting and performance meters are built into the Central Processor. These include:

Interval Timer. A Programmable source of interrupts with a range from 10 microseconds to 40.96 milliseconds

Time Base. A one microsecond relative time-of-day clock which is used by the monitor for system accounting.

Performance Analysis Counter. Designed as a tool for testing and evaluating the system, this counter monitors either duration or rate of occurrence of various hardware conditions and events.

Two Accounting Meters. An Instruction Processor Meter which measures the amount of the instruction processor time used. The information is stored in the User Process Table. While the default count is user time only, the monitor may also include interrupt time and/or Exec Mode time in this accounting. The second is a Memory Reference Meter, which measures user program accesses to core memory. This information is also stored in the User Process Table.

Trap Handling—Trap facilities exist for handling arithmetic overflow and underflow, push-down list overflow, and page failures. This trap capability is separate from the program interrupt system and can cause interpretation of the trap to be performed in either the user's or the monitor's address space as appropriate for the instruction code trapped.

Priority Interrupt System— The Priority Interrupt System is one of the most flexible that is available. Devices are assigned under program control to any one of seven priority levels through the dynamic loading of a 3-bit register within the device controller. A program can change the priority level of any device or disconnect the device from the system and later reinstate it at any other level. In the same manner, a program can set, enable, or disable any combination of levels with a single instruction. In addition, the program can assign some or all devices to the same level and generate an interrupt on any channel.

Program-assignable priority interrupt levels provide much greater flexibility than permanently hard-wired systems. An interrupt can cause the processor and the interrupting device to initiate one of several possible actions. In response to an "interrupt grant" signal, the device may supply a 33-bit word which is decoded as 18 bits address, 12 bits data, 3 bits function. The processor then does one of the following:

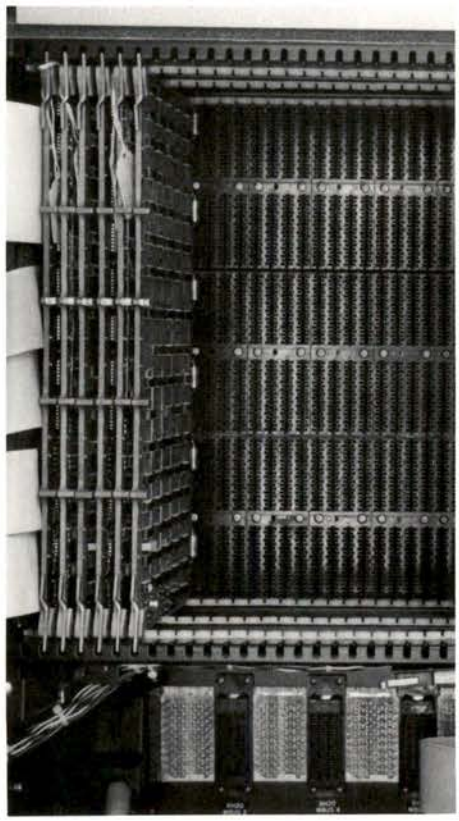
- Executes the instruction found at the supplied 18-bit address (vectored interrupt)
- Transfers a word into or out of the addressed location
- Increments or decrements the word at the addressed location

Priority level 0 is of higher priority than the seven programmable levels and is reserved for the Front End Processor. This level is invisible to the KL10 software and is provided as an access to main memory for the console.

Memory System

The Memory Control Unit contains the logic necessary for memory mapping between logical and physical addresses.

The memory address mapping hardware is integrated with the software to provide memory management totally transparent to the user. Physical memory is divided into 512 word pages. All monitor and user addresses are translated by hardware from the program's (virtual) address space to physical memory address space.



The high-order nine bits of the 18-bit virtual address define the virtual page number and are used to index into a Page Table containing the mappings between virtual addresses and physical core for either the active user or the operating system. If the physical page number is found in the 512 entry Hardware Page Table, the low-order nine bits of the virtual address are appended to it to form the complete physical address. If the virtual address is not found in the Hardware Page Table a computation is performed to determine the page mapping. The hardware page table is then loaded with the correct mapping. Two hardware registers internal to the central processor contain the page mapping information for locating the monitor's and active user's page maps in core. These maps contain information pointing to the pages for the operating system or active user, which are in core or out on the swapping device. The storage address pointers are of three basic types:

Private
the page belongs to only one user

Shared
the page is shared by more than one user

Indirect
points to another page map page where the pointer may again be one of the tree types

In addition to the physical page numbers in the page map, control bits are present to serve various functions. For example, one bit is used to indicate that the page is read only; a second is used to indicate that currently no physical page corresponds to

this virtual address slot. Another, whose function is to reduce disk channel traffic, is called a "written" bit and indicates whether or not a page need be written out to the disk or discarded when its physical space is needed.

Page level sharing among users is supported by a hardware Shared Pages Table, which holds the physical address of the shared page. This table, which is addressed through a hardware register, enables more efficient memory management because routines need to maintain just one pointer to a page, no matter how many processes are sharing it. Similarly, there is an Indirect Page Table for indirect pages.

Information about how long a page has been in core and the number of processes sharing it is also stored by the hardware in a Core Status Table to aid the monitor in memory management.

Process Tables— There are two types of Process Tables in core which are used by the software for both system and user management. These are the Exec Process Table (EPT) and the User Process Table (UPT). Each is one page long. The EPT is provided for the monitor and UPT for each user process in the system.

User Process Table

- Arithmetic overflow vector address. Overflow affects just the user and not the system, so it is handled in the user address space.
- Core and instruction processor usage accounting clocks. This information is kept for each user to aid in precise user resource accounting.

Exec Process Table

- Channel recording area. Information about channel status is maintained here.
- Front End Processor Communications Area. Used in communications between the Central and Front End Processor.

The physical memory of the DECSYSTEM-20 consists of from 64K to 256K 37-bit words. Memory increments are in 32K blocks, with up to four controllers each controlling 32K or 64K words. Two- and four-way interleaving of memory is set up through software settable switches at system startup time. Because the memories are integral to the Central Processor, memory accessing is faster than with comparable external memory architecture.

Front End Processor

The PDP-11 Front End Processor is one of the more significant features of the DECSYSTEM-20. The integration of the PDP-11 into the DECSYSTEM-20 Central Processor allows each processor to maintain its own integrity with minimal overhead. Additionally, functions are shifted to the PDP-11 wherever possible to improve overall system performance and reliability, and to provide an expandable base for future hardware and software.

Front End functions include:

- Bootstrap the Central Processor
- Drive all unit record equipment
- Service local and remote communications hardware
- Drive operator and diagnostic consoles
- Respond to Central Processor malfunctions
- Run diagnostic programs

The number of diagnostics that run under timesharing has been extended. This translates into more system availability to the user since service personnel need not take the system when repairs are needed.

To increase efficiency of service and minimize the delay that could occur if service is not onsite, a remote diagnosing capability is built in. Each system is equipped with an additional asynchronous line interface through which a remote service center, with permission of the system manager, can access the system. The service center has all the capability and diagnostic tests available as would local personnel. The result: reduced repair times and more customer satisfaction.

Diagnostic Capability — As a diagnostic computer, the Front End Processor can examine the data paths and control logic of the Central Processor even if that unit is completely inoperative. A Diagnostic Bus linkage permits automated testing procedures and allows diagnostic tests to be run that would be impossible using conventional techniques. There is less chance for human error, because maintenance engineers do not have to rely upon a malfunctioning Central Processor to diagnose itself.

The Front End Processor can sense the internal status of the central processor and control the operation of the central processor. Now, normal maintenance no longer involves time-consuming and error-prone reading of indicators and flipping of switches. All the maintenance and diagnostic procedures that require this on older computers are now performed by diagnostic programs in the Front End Processor.

Power Failure Detection and Recovery — If system power fails, a detection circuit senses the condition and causes an interrupt. The interrupt can trigger operation of a program, which saves volatile registers and provides an orderly system shutdown, allowing for later system restart with a minimum amount of lost time. A program-selectable automatic restart capability is provided to allow resumption of operation when power returns. Alternatively, a manual restart may be used.

Over Temperature Protection — Thermal sensors strategically placed within the equipment detect high temperature conditions and cause power shutdown. This, in turn, initiates the power failure interrupt.

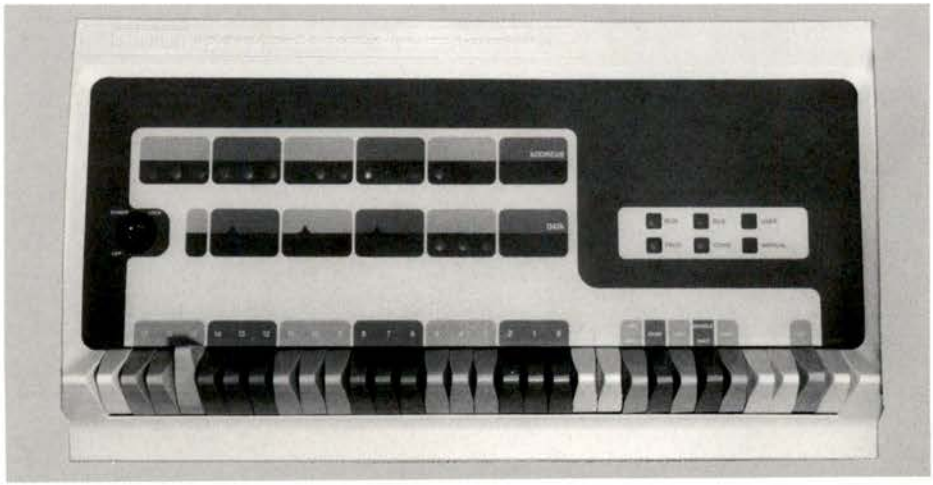
Self-Loading and System Startup — A cold start of the DECSYSTEM-20 requires a human being to push the load button and type in the time and date. It is always completed in less than five minutes. A warm restart after a temporary failure of either the DECSYSTEM-20 or the PDP-11 front end processor takes less than a minute and requires no human intervention.

DTE-20 — The DTE-20 console Front End Processor Interface connects the Central Processor and the Front End Processor to provide capabilities to interrupt, examine, deposit, diagnostic read and write, and transfer data during timesharing and diagnostic operations.

Console — The console terminal provides a direct system interface by simulating the control switch and display indicator functions for the Central Processor, thereby providing control of normal program and diagnostic operations and the ability to start, stop, load, modify, or continue a program. The console is connected for simultaneous two-way transmission through a full duplex asynchronous line interface to the Front End Processor.

Front End Processor — A PDP-11 is employed as a front end processor. It is equipped with a ROM for "cold starts", 28K words of core memory, dual floppy disks, console terminal interface, asynchronous line multiplexer, RP04 disk interface, and optionally, line printer and card reader interfaces.

The Front End Processor is equipped with a high-speed multiplexer for up to 64 asynchronous communication lines operating at speeds up to 9600 baud. The Basic System Package is equipped with a 16-channel multiplexer and 8 lines. Additional EIA compatible lines may be added in groups of eight. An optional 20mA line interface is available. The asynchronous line multiplexer meets EIA RS232C and CCITT (green book) V.24 standards.



Peripherals

A wide range of peripheral equipment is available with the DECSYSTEM-20. Each piece of peripheral equipment is fully integrated into the software system.

To increase throughput and storage capacity, a high performance disk system is provided. Removable storage elements insure maximum flexibility and convenience.

A reliable magnetic tape system operating in either of two recording modes provides data transportability and security.

To take full advantage of the interactive qualities of the DECSYSTEM-20, a high quality, high performance CRT terminal is offered.

Hard copy equipment includes DIGITAL's own widely accepted DECwriter II terminal, as well as high performance line printers in several models.

Two models of card readers are available which have been selected for their ability to work reliably with cards of varying condition.

Plotters and other applications related peripherals are available.

Today's computer systems place ever increasing emphasis on peripheral equipment and their development for the DECSYSTEM-20 is on-going.

Through this development, the user may be assured that the latest full capability peripheral equipment will be available.

RP04 Disk Systems

Up to four high-performance disk drives are supported. Each drive has a storage capacity of 100 million bytes, providing an on-line storage capacity of up to 400 million characters. On the DECSYSTEM-20, the initial disk drive is dual-ported, linking to both the PDP-11 and the DECSYSTEM-20 channels. The diagnostic and initialization routines for the system are stored on this device along with the system software.

Each RP04 disk drive transfers data directly to and from memory under the control of a channel program. Because the controller provides overlapped seeking, the operating system will concurrently position two or more drives, shortening the effective access time and increasing throughput. This feature, in combination with a rotational position sensing capability, can be used by the operating system to optimize disk throughput. A pre-load capability is provided in the controller so that when a channel program is completed the controller can send select and other control information to the disks without delay and initiate the new channel program.

The RP04 disk drive offers a high level of data reliability. A phase-locked loop clock system and state-of-the-art recording technique provide the utmost in reliable reading and recording. Additional error detection and correction hardware within the RP04 disk drive provide information on the position and pattern of any error burst up to eleven bits within the data field. Correction of most data field errors is then accomplished under software control.

The ability to offset the read/write heads facilitates read recovery and increases data reliability and improves compatibility. Moreover, if a sector on a disk pack becomes defective so that attempts by hardware and software to recover data fail excessively, the operating system dynamically marks the sector as bad so that no future attempts will be made to use it. In most cases, the data can still be recovered so that the file can be copied to a new location.

Disk System Specifications

RP04

Disk Drive Capacity	20.48 million words
Peak Transfer Rate	5.6 microseconds/word
Access time:	
Track-to-track	7 milliseconds
Average	28 milliseconds
Maximum	50 milliseconds
Organization	128 words/sector
	20 sectors/track
	19 tracks/cylinder
	411 cylinders/pack
Number of Heads	19
Number of Recording Surfaces	20
Number of Disks	11
Number of Drives/Controller	4

TU45 Tape System

Features

- 1600 bpi, 9-track
- Program-selectable recording at 1600 bpi (phase-encoded) or 800 bpi (NRZI)
- Data formats are industry-compatible
- 120,000 character/second transfer rate
- Up to 40 million characters of storage per reel
- Reading in reverse (in addition to forward)
- Expandable to eight tape drives in a single system
- Vacuum column for tape buffer
- High reliability

Up to eight dual-density tape drives may be provided. Drives are equipped with a dual gap head which enables writing and data checking on a single pass. An automatically activated erase head operates on writing.

A transport door protects the heads, capstan and magnetic tape and other tape path components from dust and other contaminants. Operator controls are accessible while the door is closed.

The drive is a single capstan unit with vacuum column tape storage system to buffer abrupt changes in velocity of the capstan from the supply and take up motors.

A tape cleaner is provided and equipped with a contaminant removal system.

The TU45 may operate in read mode in either forward or reverse direction.

The tape system is controlled by an integrated channel and controller. The operating system sets up a channel program and initializes the channel-controller. Data transfers then proceed without further process or attention.

Specifications for Tape and Control

Main Specifications	
Storage medium	½" wide magnetic tape (industry std)
Capacity/tape reel	40 million characters (at 1600 bpi)
Data Transfer speed	120,000 characters/sec., max.
Drives/control	8, max.
Data Organization	
Number of tracks	9
Recording density	800 or 1600 bits/in., program-selectable
Interrecord gap	0.50 in., min.
Recording method	NRZI for 800 bpi phase-encoded for 1600 bpi
Tape Motion	
Read/Write speed	75 in./sec.
Rewind speed	250 in/sec.
Rewind time	115 sec. typical
Tape Characteristics	
Length	2400 feet, max.
Type	Mylar base, iron-oxide coated
Reel diameter	10½ in., max.
Handling	Direct-drive reel motors, servo-controlled single capstan, filtered positive pressurized tape compartment, vacuum type tape cleaner
Error correction	(Phase-encoding only), "On-the-fly" error correction for a single-track dropout
Magnetic head	Dual-gap, read-after-write
Drives/System	8

DC20 Asynchronous Communications Multiplexer

Features

- Speeds up to 9600 baud
- 64-character hardware buffer for received characters
- DMA Transmitter for each line
- Split speed transmitter and receiver speeds may be different
- Conforms to EIA RS232-C, CCITT specifications
- Full-duplex, half-duplex

The DC20 asynchronous multiplexer is an integral part of the DECSYSTEM-20 front end. The basic system contains a 16-channel multiplexer with 8 lines implemented. Expansion to 64 lines is supported.

The system uses 16 double buffered MOS/LSI receivers to assemble incoming characters. An automatic scanner takes each received character and line number and deposits the information in a 64 character first in, first out, buffer memory referred to as the Silo.

The Transmitter also uses one double buffered MOS/LSI for each line. These are loaded directly from the front end processor's memory by means of a single-cycle, direct memory transfer. The current address and data byte count are stored in semiconductor memory within the DC20.

EIA levels with modem control leads are provided. A modem control option is available.

LP20 Line Printers

The LP20A and LP20B series printers are 300 line-per-minute, 132-column devices with 64 or 96 character print sets. The printer contains a paper advance mechanism, top-of-form control, self-test capability.

A solid state vertical format control unit (VFU), static eliminator, and paper receptacle are provided. The printer is an impact-type using a revolving character drum and one hammer per two columns. Forms with up to six parts may be used for multiple copies. Included with the printer is a control unit interfaced to the Front End Processor. The control unit operates as an NPR (DMA) device on the PDP-11 UNIBUS.

Paper and inked ribbon pass between a row of hammers and a continuously-rotating metal drum. The drum surface contains 132 columns of all print characters. Data to be printed is received and stored in a full line buffer. Printing starts when a control character (line feed, carriage return, or form feed) is sent. If more than 132 characters are sent before the control character, an automatic line feed occurs and printing continues on the next line with no loss of characters.

Printing is accomplished by scanning the stored characters in synchronization with the rotating drum characters and actuating the appropriate hammer as the desired characters move into the printing position. A 132-column line is printed in two drum revolutions; the odd-numbered columns in one revolution and the even-numbered columns in the second revolution.

High performance printers, LP20F and LP20H, are offered for applications requiring increased printer throughput. Both 64 and 96 character versions are available with scientific or EDP character sets. The full 132-column width may be easily reduced by the user so that narrower paper widths can be used.

Operation is similar to the LP20A described above, but a line is printed every drum revolution. The vertical format control is by industry standard 12 channel paper tape.

Line Printer Specifications

	LP20A	LP20B	LP20F	LP20H
Controller	LP20	LP20	LP20	LP20
Columns	132	132	132	132
Printing Characters	64	96	64	96
Printing Speed	300 lpm	240 lpm	1250 lpm	925 lpm
VFU	Prog	Prog	Tape	Tape
UNIBUS Transfer Type	NPR	NPR	NPR	NPR

CD20 Card Readers

The CD20 card readers are rated and designed to meet varying throughput requirements and provide reliable, quiet, trouble-free operation. These low-cost card readers accept 80-column EIA/ANSI standard cards.

For fast throughput, the user can choose the console model CD20B, which processes 1,200 cards/minute or the table model CD20A, which processes 300 cards/minute and has a smaller hopper while still including the same basic features as the higher speed model.

CD20 card readers are designed to prevent card jams and keep card wear to a minimum. The readers have a high tolerance to cards that have been subjected to high humidity or to rough handling and are worn or otherwise damaged.

To keep cards from sticking together, the readers use a special "riffle air" feature. The last half-inch of cards in the input hopper are subjected to a stream of air which separates the cards and air cushions them from the deck and from each other. This action unsticks those cards attracted electrostatically and loosens those cards attached through torn webs or hole locking. It also separates cards that are swollen and stuck from excess humidity.

Cards entering the reader are selected through an advanced design vacuum picker. The picker and its associated throat block prevent the unit from double picking. To minimize the chances of jamming, the card track is short (less than four inches) so that only one card at a time is in motion.

The "riffle air" and vacuum picker features greatly extend card life. Stoppages are also reduced since the reader automatically tries six times before it determines that a card cannot be picked.

The read station of the CD20 card readers uses infrared light emitting diodes as its light source and phototransistors as its sensors to provide complete reliability. No adjustments are required during the ten-year life expectancy of the diodes.

Card Reader Specifications

	CD20A	CD20B
Hopper Cap	550	2200
Stacker Cap	550	2200
Speed	300 cpm	1200 cpm
Type	tabletop	Free-Standing

LA36
DECwriter II Terminal

Features

- Fast true 30-character-per-second output
- Low cost
- Up to six-part forms
- 132-column printing
- Variable-width forms
- Upper/lower case printing
- Field-proven print head
- ANSI keyboard
- Quiet operation
- Excellent character visibility
- Mechanical simplicity for high reliability
- Meets RS232C and CCITT (green book) V.24 Standards

The LA36 DECwriter II is a modular keyboard terminal designed for high performance, quiet operation, and high reliability at speeds of 10, 15 or 30 characters per second. The LA36 provides true 30-character/second printing for full utilization of a 300-baud communications line without the use of fill characters.

Printable characters are stored in a buffer during the carriage return operation; and while more than one character is in the buffer, the printer mechanism operates at an effective speed of 60 characters/second.

Adjustable pin-feed tractors allow for variable form width from 3 to 14-7/8 inches (up to 132 columns). The print mechanism of the LA36 will also accommodate multi-part forms (with or without carbons) of up to six parts with excellent print quality.

The LA36 keyboard generates the full 128 ASCII character set, consisting of 96 upper- and lower-case printing characters and figures and 32 control characters. In addition, the LA36 is available with optional ROMs (Read Only Memories) to accommodate special character sets, in addition to the standard ASCII such as the APL character set for use with the APL Programming Language or various foreign language character sets. Characters are printed using a 7 x 7 matrix with horizontal spacing of 10 characters per inch and vertical spacing of six lines per inch. The keyboard layout conforms to the ANSI standard and is similar to a standard office typewriter. The LA36 is particularly well suited to the office environment because of its exceptionally quiet operation. To insure clear visibility of the printed line, the LA36 incorporates a beveled cabinet top for an unobstructed view, positioning of the ribbon away from the paper surface, and a print head which automatically retracts out of the way when not in operation.

The LA36 was designed for reliability and ease of maintenance. It consists of a carriage system, a ribbon feed system, a paper feed system, and an electrical system. The three mechanical systems were designed for durability and high reliability with relatively few moving parts. The electrical system is contained on two printed circuit boards, employing the latest in integrated-circuit technology.

LA36 DECwriter II Specifications

Printing	True 30-character - per - second operation; 132-column format
Spacing	10 characters per inch horizontal, 6 characters per inch vertical
Paper	Up to 6 part forms, 0.020 inches maximum pack thickness. Variable width, 3 to 14⅞ inches Tractor Drive, pin feed
Keyboard	Standard typewriter which conforms to ANSI standard X4.14-1971 typewriter paired keyboard
Power Requirements	90-132 Vac or 180-264 Vac. 47-63Hz 300 W maximum (printing)
Interface	EIA RS232C, CCITT (green book) V.24 or 20mA current mode

VT52 DECscope Video Display Terminal

Features

- Upper and lower case
- ASCII
- 24-lines
- 80 characters per line
- ANSI/typewriter compatible keyboard
- Auxiliary keypad
- Direct Cursor Addressing
- Scrolling

The VT52, Digital Equipment Corporation's newest version of the DECscope, offers a combination of features not found in any other terminal. The VT52 is an upper-and-lower case ASCII video terminal whose display holds 24 lines of 80 characters.

The VT52 is upward-compatible with the VT50, but an identification feature allows software to distinguish between the two models. Software which uses Hold-Screen Mode to produce operator-controlled, screenful-by-screenful output to the VT50 will work perfectly on the VT52 without modification despite the different screen capacities.

When a key is depressed, a subtle click occurs, giving the operator an audible response for rhythmical keystrokes. For applications where absolute quiet is necessary, this sounder can easily be disabled. Anyone familiar with a standard typewriter can learn the basics of the VT52 operation in less than an hour because the VT52 keyboard is similar to a typewriter keyboard, rather than a teletypewriter; and DIGITAL provides thorough step-by-step documentation for both programmers and operators. Errors that might occur when the operator depresses several keys in rapid sequence are prevented by the VT52's three-key rollover feature.

The VT52 provides a "two-way" auxiliary keypad. In one mode, the keypad is used to generate program-compatible numeric codes. Applications which require much numeric input can use the VT52 without modifying hardware or software, while the operator uses the convenient "numeric pad".

VT52 Specifications

Display	Format: 24 lines x 80 characters Character Matrix: 7 x 7 Screen Size: 210mm x 105mm (8.3 in. x 4.1 in.) Character Set: 96-character displayable ASCII subset (upper and lower-case, numeric, and punctuation)
Keyboard	Character Set: Complete 7-bit ASCII set (128 codes) Key layout: Typewriter—rather than keypunch—format 63 keys Conforms with ANSI standard X4.14-1971 typewriter paired keyboard Auxiliary keypad: 19 keys; numerals, cursor-movement, 3 user-definable function keys
Audible Signals	Key-click: switch-controlled Bell: Sounds (a) upon receipt of control characters BEL: (b) when keyboard input approaches right margin (output from host approaching right margin does not cause bell to ring)
Page Overflow	LF causes upward scroll: Reverse Line Feed causes downward scroll
Cursor	Type: Blinking underline Control: Up or down one line; right or left one character; home; tab (fixed tab stops every 8 spaces); direct cursor addressing (allows cursor to be moved to any character position on the screen)

Functions	Erase display from cursor position to end of line; erase to end of screen; scroll up; scroll down
Hold-Screen Mode	Allows operator to halt transmission from host, preserving data on display. Operator can request new data, line- or screenful-at-a-time. Enabled/ disabled by Escape sequences sent by system software
Terminal Self-Identification	Terminal transmits on command a sequence unique to its model; software can identify features available on any terminal it is in contact with
Communications	EIA RS232C, CCITT (green book) V.24 normally supplied on VT52 ordered with DECsystem-20 20mA current loop available Transmission rates, full duplex (switch selectable) 75, 110, 150, 300, 600, 1200, 2400, 4800, and 9600 baud Switch-selectable local copy
Dimensions	Height: 360mm (14.1 in) Width: 530mm (20.9 in) Depth: 690mm (27.2 in) Minimum Table Depth: 450mm (17.7 in)
Weight	20 kg (44 lbs)
Operating Environment	DEC STD 102-Class B environment 10°C to 40°C (50°F to 104°F) Relative Humidity 10% to 90% Maximum wet bulb 28°C (82°F) Minimum dew point 2°C (36°F)

Or, software may place the VT52 in the alternate mode, in which each key on the keypad transmits a unique Escape Sequence. This allows the host computer to distinguish between keys typed on the auxiliary keypad and similar keys on the main keyboard. In this mode, each key on the keypad can be used to invoke a user-defined function.

The VT52 has a wide range of cursor-positioning functions. As well as moving the cursor one position in any direction, software can move the cursor to any position on the screen with a Direct Cursor Addressing command which specifies the destination for the cursor. The VT52 also offers fixed horizontal tabs, a "Cursor-to-Home" command, and two screen-erasure functions. Data on the screen scrolls up when a line feed function is performed with the cursor on the bottom line; it scrolls down when a reverse line feed function is performed with the cursor on the top line.

DECSYSTEM 20 Software

The DECSYSTEM-20 is provided with highly reliable virtual memory, multi-mode operating system (TOPS-20) and a complete range of languages and supporting utilities designed to provide the user with an easy-to-learn, easy-to-use interactive system while preserving all the throughput required by traditional batch applications.

TOPS-20 simultaneously provides the features to support efficient full-language timesharing for program development and all types of interactive and terminal oriented applications, plus a full capability, multi-stream batch system.

All language processors are shareable and re-entrant to minimize memory requirements. Job control and command languages and all compilers are compatible under batch and timesharing.

The TOPS-20 operating system takes full advantage of the DECSYSTEM-20 paging hardware facilities to provide a virtual memory address space to all users of 256K words.

The system provides the concept of processes and jobs. Each job may create several processes with each process a separately schedulable entity with its own address space. Provisions are made for easy and efficient communications among processes at both the command and monitor call levels. The file system is highly developed and provides the extreme reliability so necessary in a multi-user system.

Facilities are provided for file sharing, file protection, and file backup. Files may be accessed by mapping into the user's address space. Conventional access methods are provided.

TOPS-20 is the result of the design and evolution of two other operating systems: TOPS-10, available on the large-scale DECSYSTEM-10 and proven in over 450 installations and TENEX, an advanced interactive and communications oriented system developed partly under US Government sponsorship for use with PDP-10 systems on the ARPA network.

TOPS-20 supports a full range of language processors, an easy-to-use powerful editor, a multi-stream batch system, an easy-to-learn, easy-to-use command language. It is a multi-purpose, multi-mode, virtual memory system with a highly reliable file structure.

Languages include FORTRAN, COBOL, ALGOL, BASIC, APL CPL and MACRO. Source level debuggers are provided for MACRO, COBOL and FORTRAN.

TOPS 20 Operating System

Features

- Multi-User, Multi-Mode
- Multi-Language Interactive processing
- Multi-stream batch processing
- Communications
- Virtual Memory
- Multi-process job structure
- Easy-to-learn, easy-to-use command language
- Timesharing and batch command language compatibility
- Advanced reliable file system
- Front end processor operating system for communications and unit record control

In combining two existing components to form the basis for the DECSYSTEM-20, TOPS-20 provides all the features needed for maximum system reliability and ease of use. The operating system was designed to easily and efficiently serve many user tasks but be fully protected from errors they might cause. For example, the Command Processor runs in User Mode to isolate errors caused by users interacting with the operating system. The monitor itself is divided into resident and non-resident portions to efficiently use core and logically isolate functions for enhanced reliability. Special protective hardware processor modes are utilized in the implementation.

TOPS-20 supports a large number of interactive tasks together with concurrent batch processing. Extensive human engineering has gone into the system to make it easy to use and operate. A special prompting capability within the Command Processor assists the user

by completing on request the command being entered and by identifying argument and options available. Powerful commands allow routine tasks to be performed with a minimum of typing or training.

The TOPS-20 command language is used in both batch and timesharing modes. Any terminal user can construct a job under timesharing, and then submit the job to be processed by the batch system. Concurrent with the processing of that job, he may continue interactive work under timesharing. Reliability and ease of use were conditions imposed on all aspects of the DECSYSTEM-20 software development to arrive at a product which meets the needs of today's users.

The DECSYSTEM-20 operates in a virtual memory environment. TOPS-20 operation is process oriented. Each program invoked by a user is treated as a separate process, having its own 256K word addressing space. Processes are able to initiate other processes, suspend them, and communicate with them allowing for multiple concurrent tasks to be executed. This permits modularity in application development that can be exploited at execution time. For example, an application may have three distinct phases: (1) input data, (2) process data, (3) output results. By using a separate process for each phase, it is possible to be processing one set of data while another is being input and still another is being output. For the user who doesn't need such process structuring, its support by the system is transparent to him.

Timesharing

The DECSYSTEM-20 takes maximum advantage of system throughput capabilities, allowing many independent users to share system facilities simultaneously. The conversational, rapid-response nature of the DECSYSTEM-20 makes it particularly well suited for a wide range of tasks. The system supports a variety of CRT and hard copy terminals at speeds up to 9600 baud. Terminal users can be located at the computer site or at remote locations connected to the system through asynchronous communication lines. From a terminal, the user can control the running of a program; create, edit, and delete files; compile, execute, and debug programs. The user can also request assignment of peripheral devices such as magnetic tapes. When a request for assignment is received, the operating system verifies device availability and user-access privileges. The user is then granted private use of the device until he relinquishes it.

Timesharing on the DECSYSTEM-20 is designed in such a way that the command language, input/output processing, file processing, and process scheduling are independent of the programming language used. This allows use of a wide range of languages, such as COBOL, ALGOL, FORTRAN, BASIC, APL and MACRO.

GALAXY Batch Processing System

Features

- The same command language used in timesharing
- Multi-stream operation with operator control of the number of active streams
- Automatic line printer and card reader spooling with job accounting and control, including special forms scheduling
- Job dependency scheduling allowing creation of multi-job batch applications
- Job limits parameters to define a batch job's resource bounds
- Job Control Language error recovery facilities that allow command level recovery from expected or unexpected errors
- Support of all language processors, application tools, and system utilities
- User control of automatic job restart in the event of a system failure during job execution
- CPU time guarantees for groups of users

GALAXY enables the DECSYSTEM-20 to execute batch jobs concurrently with timesharing jobs. The batch user can enter a job into the batch subsystem using a card deck which contains control cards defining the command options for the job; or, using a terminal, he can create and submit a control file which is then intercepted by the batch subsystem and processed in exactly the same manner as the job submitted on cards. Because the control language for timesharing and batch are the same, a user is

able to duplicate any terminal session with a batch job by using the same set of commands and logic. Depending on the relative proportion of batch jobs on the system or their importance, the system administrator may assign a guaranteed percentage of CPU time for batch jobs.

The batch software programs include the Input Spoolers, the Batch Controllers, the Centralized Queue Managers, Task Schedulers, and the Output Spoolers.

The Input Spooler reads from the input device and requests the queue manager to enter jobs into the Batch Controller's input queue. The input data are separated according to the control commands in the input deck and placed into either user data files or the Batch Controller's control file, for subsequent processing. In addition, the Input Spooler creates the job's log file and enters a report of its processing of the job, along with a record of any operator intervention during processing. The log file is part of the standard output for the job.

The queue manager is responsible for scheduling jobs and maintaining both the Batch Controller's input queue and the output spooling queues. A job is scheduled to run acccording to external priorities, processing time limits and parameters specified by the user for his job, such as start and deadline time limits for program execution. The queue manager enters the job into the batch input queue based on priorities. After the job is completed, the queue manager schedules it for output by placing an entry in an output queue. When the output is finished, the job's entry in the output queue is deleted by the queue manager.

The output spooling program improves system throughput by allowing the output from a job to be written temporarily on the disk for later transfer to the printer. The log file and all job output are placed by the queue manager into one or more output queues to await spooling. When the printer is available, the output is then processed by the Line Printer Spooler. The Line Printer Spooler also includes such features as special forms control and character sets, multiple copies, and accounting information.

Operator Intervention—Normal operating functions performed by programs in the batch system require little or no operator intervention; however, the operator can exercise control if necessary. He can specify the system resources to be dedicated to batch processing, limit the number of programs as well as the core and processor time for individual programs, stop a job at any point, requeue it, and change its priorities. By examining the system queues, the operator can determine the status of all batch jobs. In addition, the batch system can communicate information to the operator and record a disk log of all messages printed on the operator's console. All operator intervention during the running of the Input Spooler, the Batch Controller, and the Output Spooler causes information to be written into the user's log file, as well as into the operator's log file, for later analysis.

Job Dependency—Although jobs are entered sequentially into the batch system, they are not necessarily run in the order in which they are read because of priorities set by the user or computed by the queue manager. Occasionally, the user may submit jobs that must execute in a particular order. To ensure that such jobs are executed correctly, the user can specify initial dependency counts in the control commands of the dependent jobs. Control commands in each job on which the dependent jobs depend must in turn decrement the count; and when the count for a dependent job becomes zero, it may begin execution.

Flexibility—The batch system allows the user great flexibility. The input spooler normally reads from the card reader, but can read from magnetic tape or disk. In the command string, the user can include switches to define the operation and set priorities and limits on core memory and processor time.

Error Recovery—The user can control handling of error conditions by including special commands in his job. These commands, copied into the control file by the Input Spooler, specify the action to be taken when a program encounters a fatal error condition. If an error occurs and the user has not provided for error recovery, the Batch Controller initiates a standard dump of the user's core area and terminates the job. This dump can be used to help debug a program.

Defaults—Although the batch system allows a large number of parameters to be specified, it is capable of operating with very few user-defined values. If a user wishes to specify the nor-

mal choice for the installation, he may do so by omitting the parameters altogether. These defaults can be modified by the individual installations.

Multi-stream Capabilities— The batch system can run multiple batch jobs concurrently—a feature which enhances both system and user throughput, since a user can separate jobs into several parallel job steps.

Command Processor

The TOPS-20 Command Processor serves as the user's interface to the system. Both batch and interactive users work with the same command processor because the command language is common to both. This consistency is important because it allows switching between the two modes easily. Equally significant is the time saved to learn only a single set of commands: new user training is much simplified.

Command consistency throughout a system is quite useful, but consistency of complicated, verbose, or hard-to-remember commands is still unacceptable for the convenience and efficiency required of a computer system. TOPS-20 was designed to provide the user with concise, meaningful commands that specify what he wants done while isolating him from details better left to the system.

As an example, consider a user who has a program sample to test. In the simplest case he types EXECUTE SAMPLE. With one short command he causes TOPS-20 to compile, load, and execute his program. He doesn't concern himself with details; he doesn't pre-allocate memory disk storage or even specify where the program is to be found. TOPS-20 and the command processor cooperate to make the user's job easy. The design philosophy is that a computer should be convenient for the user—not vice versa.

To continue the example, suppose the user also wants a compilation listing of his program. To specify the listing requirement, the user types EXECUTE SAMPLE/LIST which causes the same action as before but additionally produces the desired listing.

We can carry the example further. Suppose the user is familiar with the system and doesn't like to type EXECUTE every time he wants to carry out the compile/load/execute sequence. TOPS-20 requires that the user type only enough to uniquely identify any command, so it is sufficient to type only EX SAMPLE to be equivalent to EXECUTE SAMPLE. Suppose further that the user is testing SAMPLE and goes through the EXECUTE sequence frequently. He can type only EX any time after the first EXECUTE SAMPLE, because EX uniquely identifies EXECUTE—TOPS-20 remembers and assumes SAMPLE until explicitly given a different name.

To help the user who is not fluent in all details of TOPS-20, the question mark, when typed in a TOPS-20 command, will give a list of the alternatives available. For example, typing "CH?" will make TOPS-20 type a list of all commands beginning with "CH".

Abbreviated commands are useful, but the idea is extended in TOPS-20 by allowing the user to give an abbreviated portion of a command and press the ESCAPE key. If the command is unique, the rest of the command will be typed by TOPS-20 and will include guide words to prompt the user for the arguments to the command.

Suppose a user is beginning to learn the abbreviations for commands, but has not acquired full confidence with all the abbreviations; he can type "DIR", the abbreviation for the disk directory listing command. If he has doubts that this is what he wants, typing the control character ESCAPE will make TOPS-20 type out the rest of the command that it understands. In this case the extra characters typed will complete the command line to read "DIRECTORY (OF FILES)" to indicate the full command TOPS-20 understands. This gives the user the reassurance of seeing the full command without requiring him to type it out completely.

This feature, called command recognition, works on command arguments and file names, so that whenever a user has doubts about TOPS-20's understanding of his desires he can type the control character ESCAPE and get a full description of TOPS-20's understanding of what has been commanded without actually executing the command.

The human engineering and convenience offered by the TOPS-20 command processor is a key to the power and utility of the

DECSYSTEM-20; however, the framework in which the command processor runs is equally useful. The TOPS-20 command processor is not a part of the operating system. It runs as a part of each user's job. This is possible because a job in TOPS-20 can consist of multiple tasks or processes (programs) associated with each other in a hierarchical structure. The command processor in TOPS-20 runs as the controlling process in a job.

The ramifications of this organization are many-fold. First, since the command processor is not a part of the operating system, the operating system is smaller. System operation is also more reliable because a command processor error affects only the job which exercised the bug; other jobs continue normally.

Second, the design of the system permits a job's command processor to control a task which is itself a command processor. Thus, new or special purpose command processors can be written, tested and used just like other programs, during normal system operations; stand-alone system time or operating system modifications are almost unnecessary.

Because of the program and data sharing supported on the DECSYSTEM-20, system efficiency is maintained by allowing multiple authorized users simultaneous access to a single copy of any program. Hence, the system and special command processor can be shared.

Implementation of TOPS-20 style commands and formats—recognition, abbreviation, prompting, help—is facilitated by table-oriented operating system calls (JSYS) which provide both consistency and reduced development time and effort.

In summary, TOPS-20 is engineered to be powerful yet easy to use and learn, and provides the structure and facilities required by computer users.

TOPS20 File System

Features

- Named files
- Controlled sharing among users
- Redundant root directories
- Write and close operations occur in correct order—files are always safe when reported safe
- Index Blocks kept in core for speed
- Integrated file and process space for convenience and security
- Disk and file space dynamically allocated, no preallocation required
- Complete dumping and back-up facility

TOPS-20 provides a system of named files with controlled sharing and protection among users. Data sharing is provided on a per page basis.

Reliability of the file system is ensured by design as well as by testing. File root directories are redundant. Closing of files always progresses in the correct order so that when the user is notified that a file is closed, it is known to be safe.

All file system write operations are ordered so that the directions are always correct and are left in a consistent state. When operations on a file are active, index blocks and pointers are kept in core for speed.

File and process space is integrated providing added convenience and security. This integration eliminates race conditions in transaction processing and data base applications, because unlike conventional systems, when a file is updated it is the file itself, and not a copy that is modified.

Disk and file space is dynamically allocated. There is no requirement to preallocate disk space. A complete utility is provided for backing up or dumping files. All variations of file backup procedure are available—full, incremental, and selected directories. Provision is made for implementation of a clocked dumping procedure.

Named Files

The File System on the DECSYSTEM-20 is a collection of named data sets (files) divided into 512 word pages. Each file is identified by a device, directory name, file name, and extension and generation, which defines a unique path to the file.

File Protection

All files are assigned protection codes for each of three classes of users: the owner, users within a common group or project, and all other system users. Members of each class may be granted none, some, or all of the following privileges:

- Read access
- Write access
- Permission to append data to the end of the file
- Acknowledgment that the file exists when listing the directory

When a file is created, the creator/owner may either expli-

citly assign a protection code or take the system default protection. The owner is permitted to change the protection code at any time.

Each file has associated with it a File Descriptor Block which fully describes the file's characteristics to the system. Such information includes its protection code, who the owner is, who last wrote in it, the date created, the date last changed, and its size.

File Directories

The system maintains a file directory for each user. It contains pointers to all of the user's files and file descriptor blocks as well as general information about the user. This information includes the user's password, his privileges, his disk space quota, how much disk space he has used, and the system default protection code. All directories are themselves named files, and information about them is contained in a master directory called the Root Directory. There are redundant Root Directories to increase file system integrity.

Groups

A group is a set of cooperating users established by the system administrator. Each directory contains two group membership lists: the list of users that may have group level access to the files in this Directory and the list of groups to which the owner of this Directory belongs.

Instructor/Student Example

A way in which controlled access may be used is illustrated by a typical instructor/student situation. The instructor is a member of Group A as are other instructors, but his directory may not be accessed by Group A. On the other hand, student directories are all accessible by Group A so that the instructor and all other instructors can have group level access to student files at any time, but neither students nor other instructors may access the instructor's files.

To allow for this controlled file sharing, the system offers several modes of shared access:

- Shared reading (one or more users reading, none writing)
- Writing (one user writing, none reading)
- Shared updating (one or more users reading and writing simultaneously, using the DECSYSTEM-20 queuing facility for coordination)
- Restricted updating (one user writing, any number reading)

As a further protection, if a user has opened a file for reading only, or is executing a shared program, and attempts to write in a shared page, the user who wishes to write is given a private copy of the page and the write proceeds. This "copy-on-write" facility is automatic.

All the software and hardware elements associated with the file system have been designed to provide controlled sharing of files among users on the system. Pages of core memory function as a cache between the file system and the processor.

File Backup

A system utility DUMPER provides full file backup capabilities. Capable of full or incremental file saving with provision for directory specification, DUMPER is a complete utility for saving and retrieving files.

Virtual Memory Operations

The TOPS-20 Operating System provides a demand-paged virtual memory environment. Each job on the system can be considered as a set of one or more processes, each having a unique 256K word addressing space and linked together by user options or system-defined linkages for efficient use of storage and systems resources. Operating in a demand-paged mode, the system will move into memory a subset of the total number of pages of each active process, bringing in pages from the disk or moving pages to the disk using algorithms based on the collective characteristics of all processes active on the system. A primary property of every process is its Working Set, defined as the set of pages referenced during a recent interval of time. The system determines the interval on the basis that when next the process is active, memory references will be confined with reasonable probability to the same pages. The number of process pages comprising the working set is determined by program characteristics.

A second property, definable in terms of all active processes on the system, is the Balance Set, which is the set of processes

most eligible to run and whose collective Work Sets will fit into core. Process eligibility is event driven and changes when the state of the process changes.

To insure good response, interactive processes merit a higher run eligibility than computation-bound processes. However, if any compute-bound process has been blocked by interactive processes for a long period, its eligibility is changed so as to allow it to run.

The scheduler periodically defines the Balance Set. Working Set sizes are modified dynamically as a process runs and are regularly monitored by the scheduler to adjust Balance Set membership. Administrative controls exist to guarantee a percentage of the processor to a specific user.

All paging is on a demand basis. A recently started process may begin with no pages in core. As it makes memory references, page faults occur (temporary failures due to referencing pages which are not in core) and a Working Set is built up. File input and output is accomplished by demand paging as well; the desired page is mapped into the system or user address space and is then referenced, causing a page fault.

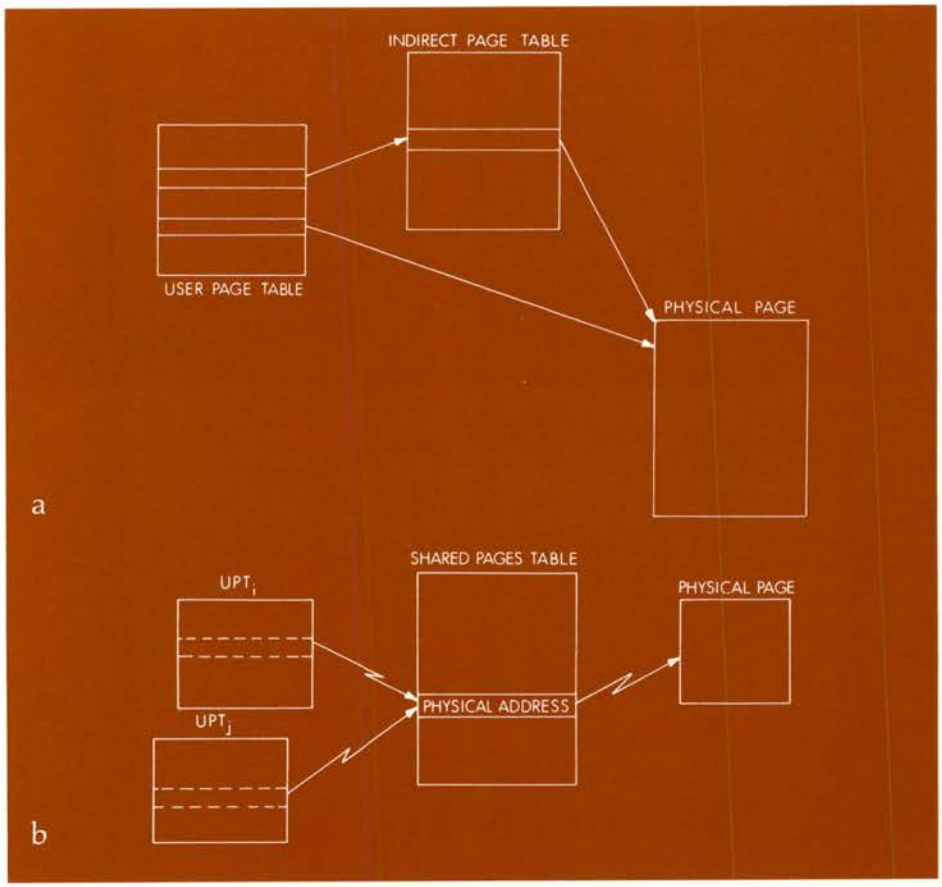
The DECSYSTEM-20 supports memory management in several important ways through hardware and microcode. Memory mapping and page level access protection are provided, so as to provide efficient sharing of pages between processes and efficient context switching. Page status and age information are automatically updated for each in core page by the microcode from information initially set up by the software. These hardware tables minimize overhead while maximizing the amount and accuracy of information available to the system for decision making in its management of core and minimization of disk channel traffic.

To effectively manage memory, the DECSYSTEM-20 uses a combination of hardware and software registers and tables to relate program or virtual addresses to physical addresses. Two hardware registers internal to the Central Processor, the EBR and UBR, contain pointers which locate the physical pages in memory containing the mapping information for the operating system and the currently active user. These pages, named the User Page Table and the Monitor Page Table, contain pointers for mapping information between the user's and monitor's address space and the actual pages of physical memory in use by the user and operating system. The pointers contained in the process tables are of three distinct types:

Immediate
Shared
Indirect

The characteristics of an Immediate Mode pointer are shown below. (a)

Entries in the User Page Table point directly to physical pages in core. Shared Mode Pointer characteristics are shown below. (b)



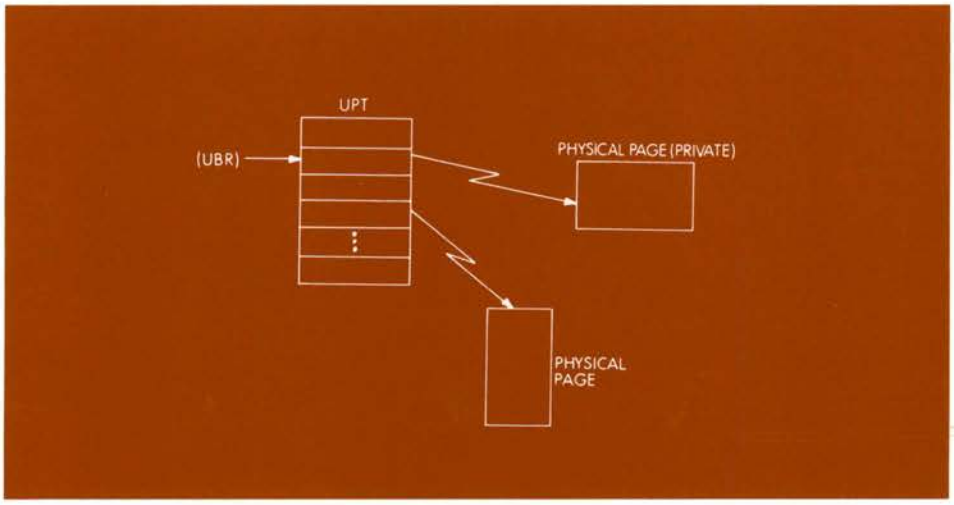
In this case, the pointer indexes into a Shared Pages Table, whose entries point to physical pages in memory. The location of the Shared Pages Table in memory is contained in another hardware register internal to the Central Processor. The advantage of having a Shared Page Table occurs when multiple processes are sharing the same page. The bookkeeping required to track the location of the shared page is reduced to updating only the Shared Page Table. Without a Shared Page Table, every user's Page Table, many of which would not be in core, would have to be modified. In instances where one process writes on a shared page, the system performs a "Copy-on-Write" action, resulting in a separate copy of page being created for the user, containing his modification. An appropriate update is then made to the information maintained in the Shared Pages Table to reduce by one the number of users sharing the specific page.

Indirect Pointer Mode is shown here:

In this mode, the entry on the Shared Pages Table does not point directly to memory, but rather points to another entry in the Process Table, which itself may re-index back into the Shared Pages Table or point directly to a physical page.

To maximize system speed, a subset of the information for all active process page mappings, including the operating system, is kept in a Hardware Page Table. As the Working Set for a particular process is generated or altered, the page mappings obtained are loaded into the Hardware Page Table. Subsequent memory reference to these pages then proceeds through the Hardware Page Table, eliminating the need to access the in-core page tables. This high-speed hardware page table minimizes the need for an "extra" cycle to pick up the required mapping information.

One additional register internal to the Central Processor points to a Core Status Table, which contains, for each page of physical core, information on recent references, page modifications, reasons for a page to be loaded in core, and reverse pointer data. The Core Status Table is used by the operating system to age pages, record when pages have been written into, record which processes have referenced the page, and to permit the operating system to locate which of the page tables point to the page.

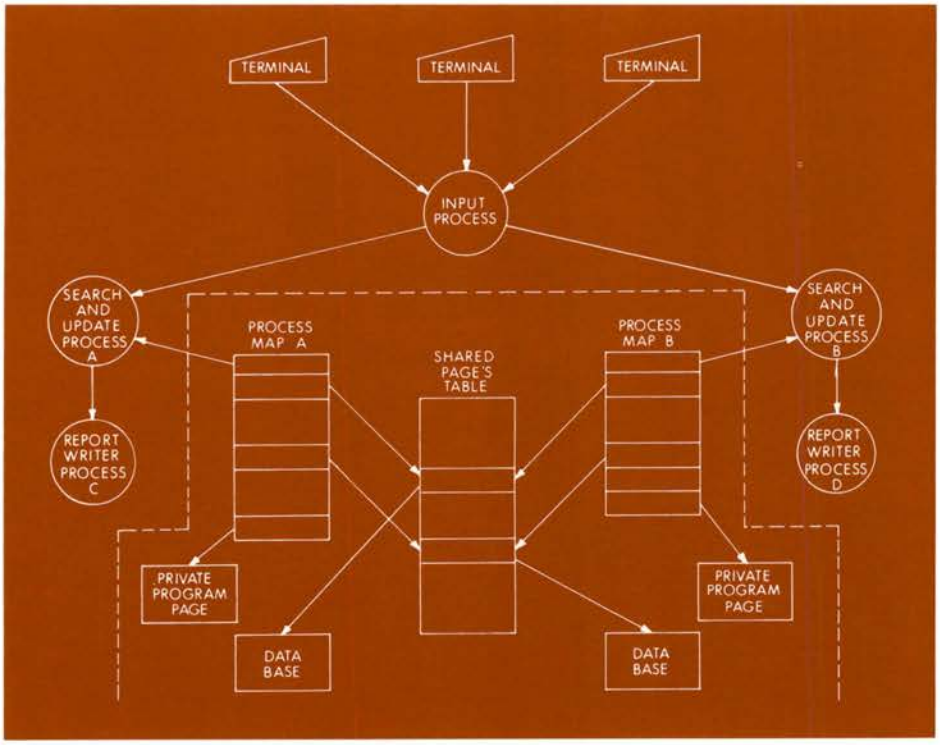


Process Structure

A process, as defined for the DECSYSTEM-20, is an independent task capable of being run by the monitor and having its own environment, including a 256K virtual address space divided into 512 pages of 512 words each. Core management is performed by the operating system, and only active process pages need be in memory for execution.

A process may create other processes which are inferior to the creating process. Two processes which have been created by another process are called parallel processes. The superior, or creating, process has control over its inferior processes and may confer or withhold privileges and suspend, continue, or terminate them.

The simplest example of a process is a single task, such as a program to solve a specific problem. It receives data, performs calculations, outputs an answer, and then exits. More complicated examples may have one task receive inputs from terminals, and pass them to subtasks for processing. These processes, in turn, may initiate other tasks to perform specific functions. The following diagram illustrates such an application.



Because each process is scheduled by the operating system individually, the system can be more responsive to the demands of the application. The ability of the system to perform a variety of functions apparently simultaneously is more easily achieved. Response time to the input terminals may be enhanced since the top level process need only receive input requests and need not be delayed due to what in another design might be a delay to update a data base.

Interprocess Communication Facility (IPCF)

Although each process is fundamentally independent, an important advantage of a multiprocess job structure is that output of some processes can be taken as input to others. The Interprocess Communication Facility allows for fast communication among processes. This communication occurs when processes send and receive information in the form of packets (messages). Each sender and receiver has a unique Process I.D. assigned to it, which is used to route packets between processes.

When one process sends information to another, it is placed into the receiver's input queue (a shared page) where it remains until the receiver retrieves it. Instead of periodically checking its input queue, the receiver can enable the software interrupt system so it can be interrupted by the sending process when the information has been sent.

To use IPCF, the system administrator must assign a user two quotas which designate the number of sends and receives his process may have outstanding at any time. For example, if a user has a send quota of two and has sent two messages, he cannot send any more until at least one has been retrieved by its receiver. A user cannot use the facility if his quotas are zero.

Software Interrupt System

The software interrupt system provides a mechanism for a process to respond to several types of interrupts generated by other processes, hardware, error conditions, and terminal interactions. It may be enabled by both the user and the system. The system provides 36 software interrupt channels, of which 18 have reserved functions. The primary types of interrupts are:

- Terminal character interrupts
- IPCF interrupts
- ENQ/DEQ interrupts
- Hardware error conditions interrupts
- I/O error condition interrupts
- Program errors, such as arithmetic overflow
- Interprocess interrupts

There are three interrupt priority levels assignable to each software interrupt channel, thus providing priority ordering. A call is provided for interrupt

dismissal to resume the interrupted code.

Enqueue/Dequeue (ENQ/DEQ)

Although TOPS-20 makes program and data sharing very easy, while preventing any user from changing a file which others are using in read-only mode, more complex applications require simultaneous updating of a data base (file) or sharing of devices.

By using ENQ/DEQ facility, cooperating processes can insure that such resources are shared correctly and that one user's modifications do not interfere with another's. Examples of resources that can be controlled by this facility are devices, files, operations on files (e.g., READ, WRITE), records, and memory pages.

Front End Software

The Front End Processor and software reduce the central processor's participation in I/O operations and provide an extensive diagnostic/maintenance tool.

The purpose of the Front End Processor is to reduce the workload of the Central Processor and to serve as a powerful diagnostic and maintenance tool for service personnel. The following functions are carried out by the Front End Processor:

- Console processor
- Communications processor
- Peripheral processor
- Diagnostic/Maintenance processor

Console Processor—Two major functions of the console processor are system initialization and system-operator communications. A person need only push a button and type in the date to initiate the following automatic program sequence. The Front End Processor loads and verifies the microcode, configures and interleaves memory, and loads and starts execution of the Central Processor bootstrap program from a device specified by the user.

The user may request a memory configuration list indicating which memory controller is on line, what is the highest memory address configured, and how the memory is interleaved.

All normal console capabilities, such as lights, start, stop, reset, and load, are provided by the Front End Processor. An operator command language is provided with the system.

Communications Processor—The Front End Processor serves as an intelligent data link and buffer for the interactive terminals, thereby relieving the Central Processor of this overhead. The Front End Processor buffers the characters received and interrupts the Central Processor upon a clock signal or when it has a group of characters to send. The Central Processor also sends groups of characters for terminal output to the Front End Processor, further enhancing efficiency. Programmable terminal speed setting capability is provided, allowing for terminal speeds to be changed dynamically.

Peripheral Processor—The unit record spooling programs in TOPS-20 communicate with the Front End Processor by the same buffering mechanism described for the communication processor. Both the Central Processor and the Front End Processor maintain buffers for data, and interrupt only once per buffer transmission, thus increasing the amount of useful work each processor can do.

Diagnostic/Maintenance Processor—An integral part of the DECSYSTEM-20 is a remote diagnosis capability that reduces Mean Time To Repair (MTTR) and increases system availability to the customer.

This capability allows Service Engineers to determine the cause of most system failures before leaving the local office. By running diagnostic tests using telephone dial-up facilities, the Field Service Engineer can give the customer better service because he can better select which spares and test equipment should accompany him on the call. Most tests may be run during general timesharing. Even if the main CPU is inoperative, the Front End Processor can access all busses and major registers.

Total system security is maintained because the on-site system operator must enable the communications link, and only the on-site personnel may specify the specific password to the system each time remote diagnosis is employed. Time limits for system access may also be imposed, and the remote link may never operate at a higher privilege level for commands than the local console terminal. All input and output to the diagnostic link is copied to the local terminal so that on-site personnel have a record of all steps taken.

System Reporting and Control

One of the major system design goals for the DECSYSTEM-20 has been to provide a high degree of control and simplicity of operation. Areas of specific concern include:

Accounting and Administration—Files are maintained by the system to provide information on resource usage (CPU, disk space, etc). This provides the basis for charging individual users as well as providing a year-to-date summary and usage statistics.

The system administrator sets resource allocation guidelines: disk quotas for each individual user to control disk usage, the amount of resources for batch and timesharing, and the maximum sizes of spooled output files.

Certain privileges can be granted to key personnel. These could cause an unnecessary loss of system security and reliability if used incorrectly. To further protect the system and the privileged user, his privileges are only in effect if he specifically enables them.

System Generation—The monitor is self-configuring.

Error Reporting—The TOPS-20 Operating System incorporates an extensive error detection and recovery package to insure maximum system availability to the user. It also incorporates complete error recording facilities for use by DIGITAL maintenance engineers and support personnel, and customer operating staff.

When an error is detected, the TOPS-20 monitor gathers all pertinent hardware and software information. The recovery procedure is then started and error related information is added to a disk file for later use.

The DECSYSTEM-20 is provided with an extensive range of user oriented high-level languages. The development of high-level language has been an important process in the evolution of improved computer systems.

The solution to problems can best be expressed in the terms most closely related to the problem. The scientific user has come to depend upon FORTRAN, ALGOL and APL. Commercial users apply COBOL to business data processing tasks.

For the beginning or casual user, BASIC is often the first choice. Interestingly, many useful and extensive applications packages have been developed in the BASIC language.

Just having language translators is not sufficient. The DECSYSTEM-20 user's work has been made easier by providing extensions to standard languages and by relaxing certain restrictions.

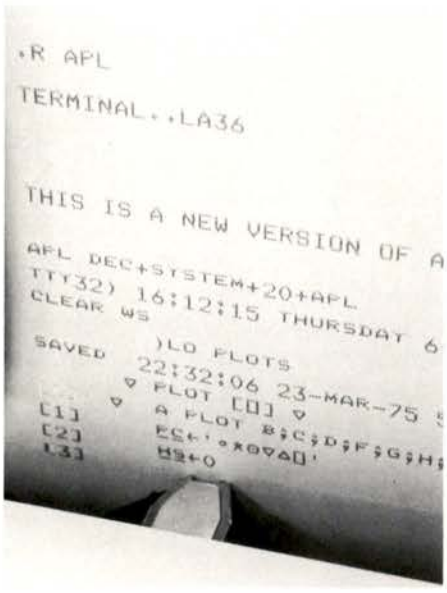
Better language implementations permit the computer to handle more of the annoying details while the user can spend more of his time solving problems.

Because writing a program is only one step in the program development process, the DECSYSTEM-20 provides not only a superior interactive facility for entering the program, but also, superior debugging tools which operate at the source program level.

DIGITAL originated the interactive debugging concept when it was first to introduce DDT, (Dynamic Debugging Technique) variants of which are now offered on other systems. To make the usefulness of interactive debugging available to a wider range of people, the DECSYSTEM-20 FORTRAN and COBOL have source level debugging facilities—FORDDT and COBDDT.

These tools help to maximize the user's efficiency through the assistance they provide in getting his program operational.

To ensure that programs run as efficiently as possible, DECSYSTEM-20 produces efficient code taking advantage of the program's structure and the superior instruction set of the DECSYSTEM-20. To minimize core requirements DECSYSTEM-20 compilers are re-entrant and may produce re-entrant code.



ALGOL

- Programs may comprise separately-compiled procedures
- Assignments are permitted within expressions
- Remainder operator
- Unique implementation of dynamic own arrays
- Octal boolean constants and integer/boolean and boolean/integer transfer functions
- Alternative reserved delimiter word or "quoted delimiter word" representations

The DECSYSTEM-20 ALGOL system is based on the ALGOL-60 specifications. It is composed of the ALGOL compiler, and the ALGOL object time system. The compiler is responsible for reading programs written in the ALGOL language and converting these programs into machine language The user may specify a parameter at compile time which produces a sequence numbered listing with cross-referenced symbol tables.

The ALGOL object time system provides special services, including the input/output service for the compiled ALGOL program. Part of the object time system ALGOL library is a set of routines that the user's program can call in order to perform various functions including mathematical functions and string and data transmission routines. These routines are loaded with the user's program when required; the user need only make a call to them. The remainder of the object time system is responsible for the running of the program and providing services for system resources, such as core allocation and management and assignment of peripheral devices.

Source level, interactive debugging is provided through ALGDDT. The user may stop his program at any point, may examine and change the contents of data locations, may insert connected code and then continue execution to test his changes.

APL

APL (A Programming Language) is a concise programming language specially suited for dealing with numeric and character data which can be collected into lists and arrays. The conciseness with which APL expressions can be written greatly enhances programmer productivity and allows for very compact and readable code which can be executed in a highly efficient manner. APL finds application not only in mathematics and engineering, but also in financial modeling and text handling situations.

Some of APL's features include:

- The user's active workspace size is dynamically variable
- The workspace symbol-table is dynamically variable
- Immediate-mode line editing of complex APL expressions
- Statement branching may occur anywhere in a statement line
- Multiple statements may appear in a single line
- User-controlled tab positioning for I/O operations
- All floating point operations are performed to an accuracy of eighteen decimal digits

APL is a fully interactive system with both immediate (desk calculator) and function (program) modes of operation. It includes its own editor as well as debugging tools, tracing of function execution, typeout of intermediate values, setting break points, etc. APL for the DECSYSTEM-20 includes many extensions not normally found in other APL implementations. Some of these features include a flexible file system which supports both ASCII sequential and binary direct access files, the dyadic format operator, expanded command formations which allow the user to take advantage of the DECSYSTEM-20 file system, the execute operator and tools for error analysis and recovery. In addition, workspaces are dynamically variable in sizes up to 512K bytes and the system supports double precision calculations which allow up to 18 decimal digits of precision.

Finally, APL on the DECSYSTEM-20 is available to all users at all times without the need to run a separate subsystem which provides extra terminal handling, workspace swapping, etc. These tasks are all accomplished by the TOPS-20 operating system.

BASIC

- Multiple users share the BASIC compiler's pure code
- A variety of program manipulation commands which include functions for saving, running and retrieving BASIC programs
- Immediate mode statements to facilitate debugging and "desk calculator" mode
- Sequential data storage
- String capability; including string arrays and functions
- Chaining with COMMON to accommodate large programs
- Formatted output using the PRINT USING statement

BASIC is a conversational problem solving language that is well suited for timesharing and easy to learn. It has wide application in the scientific, business, and educational communities and can be used to solve both simple and complex mathematical problems from the user's terminal.

The BASIC user types in computational procedures as a series of numbered statements that are composed of common English terms and standard mathematical notation. After the statements are entered, a run-type command initiates the execution of the program and returns the results.

If there are errors during execution, the user, from his terminal, simply changes the line or lines in error by deleting, modifying, or inserting lines.

The beginning user has many facilities at his disposal to aid in program creation:

- **Program Editing Facilities**—An existing program or data file can be edited by adding or deleting lines, by renaming it, or by resequencing the line numbers. The user can combine two programs or data files into one and request either a listing of all or part of it on the terminal or a listing of all of it on the high-speed line printer.
- **Documentation Aids**—Documenting programs by the insertion of remarks within procedures enables recall of needed information at some later date and is invaluable in situations in which the program is shared by other users.
- As a BASIC user, you can type in a computational procedure as a series of numbered statements by using simple common English syntax and familiar mathematical notation. You can solve almost any problem by spending an hour or so learning the necessary elementary commands.
- **Functions**—Occasionally, you may want to calculate a function, for example, the square of a number. Instead of writing a program to calculate this function, BASIC provides functions as part of the language.

More advanced users will want to take advantage of some of the more sophisticated computation and data handling tools which BASIC provides. For example:

File Input—The user may create a name date file using the BASIC editing facilities. This file may now be referenced within a program.

Output Formatting—The user can control the appearance of his output to the terminal or printer.

Data Access—Data files may be read and written either sequentially or randomly.

String Manipulation—Alphanumeric strings may be read, printed, concatenated and searched.

COBOL

The COmmon Business Oriented Language, COBOL, is an industry-wide data processing language that is designed for business applications, such as payroll, inventory control, and accounts receivable.

Because COBOL programs are written in terms that are familiar to the business user, he can easily describe the formats of his data and the processing to be performed in simple English-like statements. Therefore, programmer training is minimal, COBOL programs are self-documenting, and programming of desired applications is accomplished quickly and easily.

Major features include:

- **On-line editing and debugging**
  The programmer may create and modify the source program from a terminal using an easily-learned editing facility.
  After the program has been submitted to the compiler—again from the terminal—any syntax errors may be immediately corrected using the editor, and the program resubmitted. The program listing is sequence numbered with symbols cross-referenced to show when they are defined and where used. COBDDT, the on-line, interactive COBOL debugging package, allows the programmer to check out the program:
  - By selectively displaying a paragraph name
  - By causing the program to pause at any desired step during execution
  - By allowing the program to examine and modify data at will before continuing execution

  Easy program development and debugging increases programmer productivity by eliminating tedious waiting periods
- **Batch**
  Using the same commands for timesharing, the programmer may submit the program via a control file to the Batch system. This further increases efficiency because other terminal work may be done concurrent with the Batch processing
- **Access Methods**
  The programmer has a choice of three access methods; sequential, indexed sequential, and random.

Indexed Sequential Access Mode (ISAM)

ISAM requires a minimum amount of programming while providing a large data file handling capacity. It is supported by the COBOL Object Time System which automatically handles all of the searching and movement of data.

All reading and writing of an index file (ten levels of indexing) are performed by the run-time operating system (LIBOL). This does not involve the user. When using the indexed sequential files, the programmer need only specify which record is to be read, written, rewritten, or deleted.

Whenever records are added to the file, the index is automatically updated; additions to the file will not degenerate the file as with other computers. The common technique of using overflow areas for added records has been avoided. Whenever records have been deleted from the file, the empty space is used again for later additions. The net effect of the addition and deletion techniques significantly increases the time between major "overhauls" of the data files, because the time required to access a file is independent of the number of changes made to it.

- **Sorting**
  The SORT package permits a user to rearrange records or data according to a set of user-specified keys. The keys may be ascending or descending, alpha or numeric, and any size or location within the record. Multi-reel file devices may be specified.
- **Source Library Maintenance System**
  This system lists file entries or adds, replaces, and/or deletes source language data on a file. It can also add, replace, or extract an entire file from the library.
- **Device Independence**
  The operating system allows the programmer to reference a device with a user-assigned logical name as well as its physical name, thus allowing dynamic assignment of peripheral devices at run time.

CPL (Conversational Programming Language)

An interpreter supporting a subset of the draft ANSI PL/1 language.

CPL is intended to be easy-to-use for the beginning programmer or non-programmer. At the user's option, statements will be executed immediately or saved for deferred execution. A beginning programmer can start by executing simple computational statements and can proceed to building programs.

Since CPL is an interpreter, a user can track a program very closely. Debugging features, such as source level breakpoints and program modifications, are available.

CPL supports the data types FIXED, FLOAT, CHARACTER, and BIT. Most standard PL/1 functions are supported.

- **Data Base Management System Interface**
  The programmer may call upon the data base management system facilities from within his program. This system, which follows the CODASYL specifications, allows data files to be consolidated into one or more data bases. Application programs are then permitted to access the data in the way best suited to their needs.

FORTRAN

The FORmula TRANslator language, FORTRAN, is a widely-used and procedure-oriented programming language. It is designed for solving scientific problems and is thus composed of mathematical-like statements constructed in accordance with precisely formulated rules. Therefore, programs written in FORTRAN consist of meaningful sequences of these statements that are intended to direct the computer to perform the specified computations.

FORTRAN has a wide use in every segment of the computer market. Universities find that FORTRAN is a good language with which to teach students how to solve problems via the computer. The scientific community relies on FORTRAN because of the ease with which scientific problems can be expressed. In addition, FORTRAN is used as the primary data processing language by many timesharing utilities.

FORTRAN Version 4

FORTRAN-20 is a superset of the American National Standard FORTRAN. Both the compiler and object-time system are reentrant (shareable). The compiler produces optimized object code. The following is a summary of key features and extensions of FORTRAN-20:

- PARAMETER statement—Allows symbolic specification of compile-time constants
- INCLUDE statement—Allows user to include in the compilation of a given program unit source code which resides on a file other than the primary source file
- GLOBAL OPTIMIZATION-FORTRAN-20 performs extensive local and optional global optimizations
- FORDDT—the FORTRAN-20 debugger
  FORDDT in conjunction with the FORTRAN-20/"DEBUG" switch will allow:
  - Display and modification of program data
  - Tracing of the program statement by statement
  - Setting of pauses on any statement or routine
- OPEN/CLOSE file-specification statements
- Array bounds checking can be generated optionally
- N-dimensional arrays
- ENCODE/DECODE statements
- Boolean operations equivalence (EQV) and exclusive or (XOR), in addition to OR, AND, NOT.
- The NAMELIST and list-directed I/O features provide format-free input and output operations
- Random-access I/O capabilities
- Compatibility with IBM type declaration statements
- Implied DO loops in I/O statements and data statements
- Full mixed-mode arithmetic in expressions
- Octal Constants

- Logical Operations—full-word, masking operations for all logic functions (rather than a result of just true or false)
- END= and ERR= in I/O statements
- Device independence

The FORTRAN-20 object-time system, FOROTS, controls the input/output, format interpretation and numerical conversion for compiled programs. The FORTRAN user may reference any I/O device. All special editing, conversion, and file structuring tasks are handled by the object-time system. Devices are normally specified by logical assignment so that physical device selection need not be made until run-time. The devices corresponding to the special I/O statements READ, PRINT, ACCEPT, and TYPE are also assignable at run-time. The object-time system is shareable.

FOROTS implements all program data file functions and provides the user with an extensive run-time error reporting system. Device independence is provided to allow the programmer or operator to determine the physical device at run time.

FORTRAN-20 is easy to use in both timesharing and batch processing environments. Under timesharing, the user operates in an interactive editing and debugging environment. FORDDT, an interactive program that is used as an aid in debugging FORTRAN programs uses the language constructs and variable names of the program itself—thereby eliminating the need to learn another language in order to accomplish on-line debugging. Under batch processing, the user submits his program through the batch software in order to have the compiling, loading, and executing phases performed without his intervention.

FORTRAN programs can be entered into the FORTRAN system from a number of devices: disk, magnetic tape, user terminal and card reader. In addition to data files created by FORTRAN, the user can submit data files or FORTRAN source files created by the system editor. The data files contain the data needed by the user's object program during execution. The source files contain the FORTRAN source text to be compiled by the FORTRAN compiler. Output may be received on the user's terminal, disk or magnetic tape. The source listing cross references all symbols to show where they are defined and where used.

MACRO

MACRO is the symbolic assembly language on the DECSYSTEM-20. It makes machine language programming easier and faster for the user by:

- Translating symbolic operacodes in the source program into the binary codes needed in machine language instructions
- Relating symbols specified by the user to stored addresses or numeric values
- Assigning relative core addresses to symbolic addresses of program instructions and data
- Providing a sequence numbered listing with symbols cross-referenced to show where they are defined and where used

MACRO programs consist of a series of free format statements that can be prepared on the user's terminal with a system editing program. The elements in each statement can be entered in free format. The assembler interprets and processes these statements, generates binary instructions or data words, and processes a listing which may contain cross reference symbols for ease in debugging. MACRO is a device-independent program; it allows the user to select, at run-time, standard peripheral devices for input and output files. For example, input of the source program can come from the user's terminal and output of the assembled binary program can go to a magnetic tape, and output of the program listing can go to the line printer. More commonly, the source program input and the binary output are disk files.

The MACRO assembler contains powerful macro capabilities that allow the user to create new language elements. This capability is useful when a sequence of code is used several times with only certain arguments changed. The code sequence is defined with dummy arguments as a macro instruction. Thus, a single statement in the source program referring to the macro by name, along with a list of the real arguments, generates the

entire sequence needed. This capability allows for the expansion and adaptation of the assembler in order to perform specialized functions for each programming job.

DDT

The on-line symbolic debugging tool, DDT (Dynamic Debugging Technique), enables the user to perform rapid checkout of a new program by making a change resulting from an error detected using DDT and then immediately executing that section of the program for testing and loading; the user may enter DDT at any time, either before execution or after an error has occurred. After the source program has been assembled, the binary object program, with its table of defined symbols, is loaded with DDT. Through command strings to DDT, the user can specify locations in his program, called breakpoints, where DDT is to suspend execution in order to accept further commands. In this way, the user can check out his program section-by-section and, if an error occurs, insert the corrected code immediately. Either before DDT begins execution or at breakpoints, the user can examine and modify the contents of numbered text modes. DDT also performs searches, gives conditional dumps, and calls user-coded debugging subroutines at breakpoint locations. The important feature of DDT is that user communication with DDT is in terms of the original symbolic location names, rather than the machine-assigned locations.

EDIT

Edit allows on-line creation and modification of programs and data files. The user may insert, delete, or print lines, as well as modify lines without having to retype them. Advanced techniques include string searches, substitutions and text manipulations.

The novice finds the system easy to use due to the user-oriented syntax and extensive helping facilities. At the same time the more advanced user may perform quite intricate text manipulation with equal ease.

Dumper

Dumper provides file backup for the disk file structure. Installations may use it to dump and restore all files in all directories, to dump and restore only those files which have been changed since last dump, and to dump and restore individual user directories.

LINK

LINK, the DECSYSTEM-20 linking loader, merges independently-translated modules of the user's program and any necessary system modules into a single module that can be excuted by the operating system.

The primary output of LINK is the executable version of the user's program. The user can also request auxiliary output in the form of map, log, save, symbol, overlay plot, and expanded core image files by including appropriate instructions in his command strings to LINK. The user can also gain precise control over the loading process by setting various loading parameters and by controlling the loading of symbols and modules. Furthermore, by setting parameters in his command strings to LINK, the user may specify the core sizes and starting addresses of modules, the size of the symbol table, the segment into which the symbol table is placed, the messages he will see on his terminal or in his log file, and the severity and detail of any error messages. Finally, he can accept the LINK defaults for items in a file specification or he can set his own defaults that will be used automatically when he omits an item from his command string.

RUNOFF

RUNOFF is the DECSYSTEM-20 documentation preparation program. It provides line justification, page numbering, titling, indexing, formatting, and case shifting. The user creates a file which includes text and information for formatting and case shifting, with an editor. RUNOFF processes the file and produces the final formatted file to be output to the terminal, the line printer, or to another file.

With RUNOFF, large amounts of material can be inserted into or deleted from the file without retyping text that remains unchanged. After a group of modifications has been added to a file, RUNOFF produces a new copy of the file which is properly paged and formatted.

SORT

The TOPS-20 SORT arranges the records of one or more files according to a user-specified sequence. The user designates the keys on which the records are sorted from one or more fields within a record. The keys can be in either ascending or descending order. SORT compares the key fields values of all records. Then it arranges the records in the specified sequence and merges them into a single output file. Sort may be used under timesharing and batch, and may be called from within a COBOL program.

Data Base Management System

Certain data—such as that within commercial, accounting, inventory control and administrative systems—are used in computer applications which have common relationships and processing requirements with data in other applications. This can prove to be a problem because as file organizations are defined in one application or program, restructuring, redundant appearance, and even repetitive processing of the same data are often required in another application.

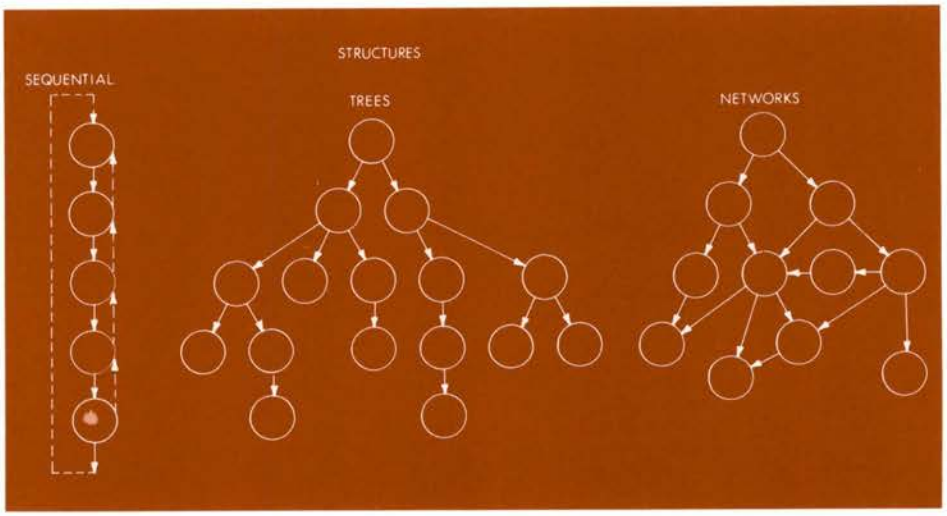
To further complicate the problem, on-line processes (i.e., customer order entry, shipment planning, and student information retrieval systems) tend to different data structures than the processes used to create and maintain primary data files. This means that as new applications requirements for existing data are determined and additional data are defined, program development personnel must either alter existing data file forms and programs or create and maintain redundant copies.

DIGITAL offers a solution to the problem in the Data Base Management System—DBMS. It enables DECSYSTEM-20 users to organize and maintain data in forms more suitable to the integration of a number of related but separate processes and applications. DBMS is ideal in situations where data processing control and program development functions require structures and techniques not satisfied by traditional data management facilities.

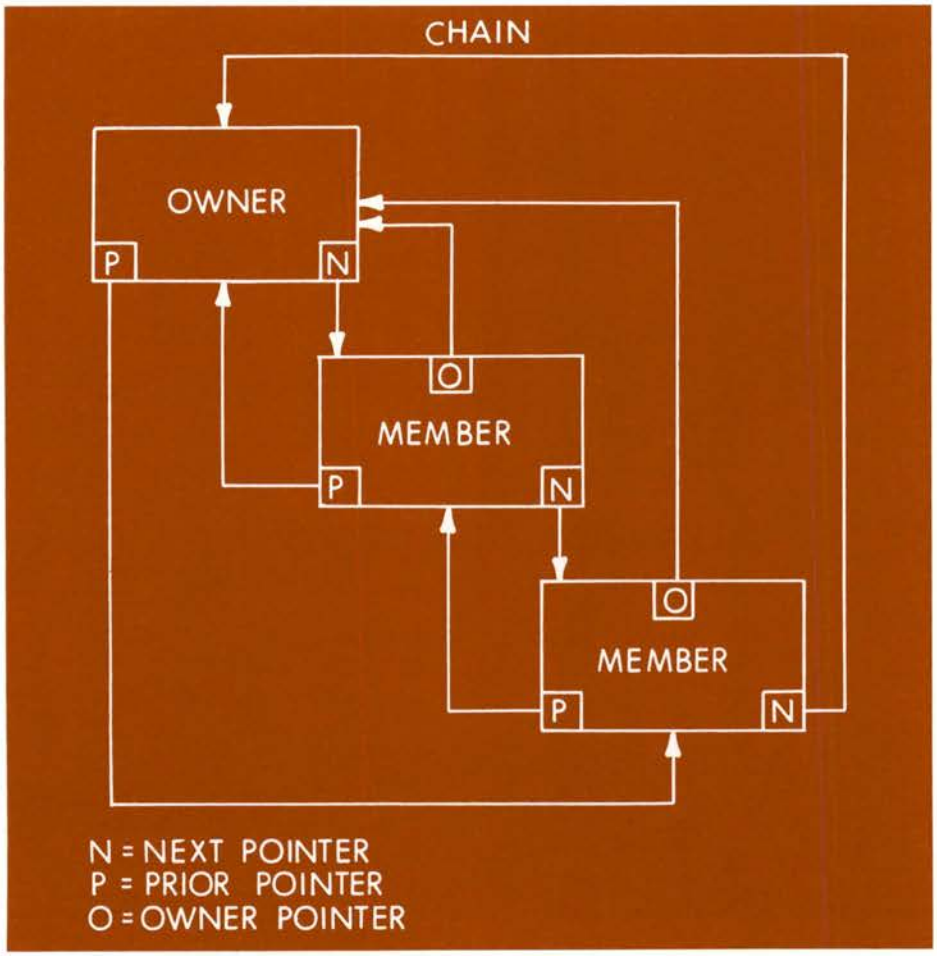
Features

The DECSYSTEM-20 Data Base Management System is predicated upon the proposals of the CODASYL Data Base Task Group (DBTG) which appear in their report of April, 1971. DIGITAL's goal for DBMS is to provide features which assist users in obtaining the most significant objectives stated in the CODASYL DBTG report. These features include:

- **Hierarchical Data Structures** —In addition to sequential structures, simple tree structures and more complex network structures can be created and maintained. Data items can be related within and between various levels of the structure established.



- **Non-redundant Data Occurrence**—Data items may appear in a number of different structural relationships without requiring multiple copies of the data. Data structures may be established and modified in a manner most suitable to a given application without altering the occurrence of the data in structures maintained for other applications.
- **Variety of Access and Search Strategies**—Data records can be maintained in chains, as illustrated below. Access may be through DIRECT, CALCULATED, or VIA set location modes. Sort keys, in addition to the normal storage key, may be defined.
- **Concurrent Access**—Multiple run units (or programs) that use the same re-entrant code module further supplement the DECSYSTEM-20 monitor's extensive centralized file handling capabilities. Any number of concurrent retrievals to the same data areas can be handled. Concurrent updates to the same area can be handled by a multiple update queuing mechanism.
- **Protection and Centralized Control**—Usage of a common data definition language establishes data base structures maintained on direct access storage devices that are selectively referenced by individual application programs through privacy lock and key mechanisms. Physical placement of data is specified by a centrally-controlled data definition processor.
- **Device Independence**—The common input/output and control conventions of the DECSYSTEM-20 monitor provide basic device independence. Applications programs deal with logical areas rather than physical devices. Data base areas may reside on the same or different direct access storage devices as non-date base files.
- **Program Independence of Data**—Significant steps toward program independence are achieved by using the SCHEMA and SUB-SCHEMA concepts of data definition. Individual programs reference only user-selected data elements rather than entire record formats. Program alterations due to adding new data and relationships are minimized. Alterations of the original form (i.e., binary, display) and element sizes require program recompilations.
- **User Interaction Without Structural Maintenance Responsibilities**—Individual users, applications, and programs may access data structures without the responsibility of maintaining detailed linkage mechanisms that are internal to the data base software. Common and centralized authorization, error recovery, and building techniques reduce individual activity to maintain data structures.
- **Multiple Language Usage**—The DBMS data manipulation language is available to



Services

both COBOL and FORTRAN as host languages
- DBMS Software Modules

Consistent with the CODASYL Data Base Task Group report, the body of DECSYSTEM-20 data base management software includes:

- DDL—data description language and its processor
- DML—data manipulation language for COBOL and FORTRAN programs
- DCBS—data base manager module of re-entrant run-time routines
- DBU—group of data base system support utilities

Quality in DIGITAL computer products is acknowledged throughout the world. Our engineers and programmers have provided reliable computer hardware and software to solve a host of problems in a variety of fields. To complement our product offering, we have developed a comprehensive customer service capability.

DECSYSTEM 20 Maintenance

Field Service offers total maintenance capability to DIGITAL customers. Our service engineers are experts in computers. They typically have over eight years experience in computer systems and are your assurance of quick and correct field maintenance. You receive their assistance (and greater system productivity) through the DIGITAL Service Agreement.

Planning for adequate service is vital. Your computer system usage determines the scope of your needs. The DIGITAL Service Agreement is structured to meet those requirements. Characteristics of the agreement include:

- All parts and labor
- Priority response
- Guaranteed availability
- Preventive maintenance
- Spare parts inventory
- Tailored coverage

These provisions help lead to greater productivity from your DECSYSTEM-20.

Logistics

Good service demands outstanding logistics. Spares availability minimize computer repair time. Our materials inventory network recognizes this concept—and strengthens the total DIGITAL support system.

The first level of spares is often at your own computer site. From this point the inventory is in echelons at the Branch, District, Regional and Headquarters locations. Our own DECSYSTEM-20 keeps track of this support system, providing replacement parts when needed.

For DIGITAL computers located in areas on the fringes of our service capabilities, and for customers who use our maintenance on an "as available" basis, DIGITAL supplies spares planning and inventory procurement assistance.

DECSYSTEM 20 Training

Education

Education is an integral part of the total DIGITAL customer service system. We believe in training, not only for ourselves, but also for our customers. Your employees will, through education, extract greater performance from your computer.

To provide this training, we have established completely equipped training centers in a dozen locations around the world. Our staff at these centers consists of full-time instructors dedicated to computer training.

The education DIGITAL offers includes standard and custom courses in both hardware and software. Our current schedule

includes over 100 standard courses, ranging in duration from one to five weeks.

DIGITAL recognizes that you may have a unique requirement that can best be met with education at your location. Our on-site program meets this need by designing and conducting courses where and when you choose.

Types of Courses

Catalog Courses— Educational Services publishes its "Educational Courses Catalog" semi-annually. This catalog includes a full listing of hardware and software courses geared to DIGITAL's products, as well as many generic computer science courses. One section of this catalog is devoted to DECSYSTEM-20 training. Courses are listed with their schedules for the current six (6) month period. Prerequisites, course abstract, objectives and topic outlines are included.

Seminars— Conducted at our Regional Education Centers and at other convenient locations, seminars are designed to complement and expand upon our existing curriculum, as well as keep the customer informed of DIGITAL's product developments. By bringing these workshops to the customer's locale, Educational Services furnishes a convenient way for the DECSYSTEM-20 customer to gain additional information on his system.

On-Site Courses— If the customer has a group of individuals to train, it is often more economical to have Educational Services conduct courses at the customer's location rather than one of our Education Centers.

Custom Courses— If the customer has a unique application or training situation, custom courses are very effective. This approach gives the customer the added advantage of maximizing relevant material content while minimizing information extraneous to his operations.

DECSYSTEM 20 Training Program

Training Credits— Along with the purchase of a DECSYSTEM-20 the customer will receive ten (10) training credits. Training credits may be applied to the standard software courses for a period of one year from the date of issue.

If used as suggested below, the ten training credits will enable one system programmer, one system administrator, and one applications programmer to attend the appropriate software courses in Marlborough free of tuition charge:

System Programmer
DECSYSTEM-20 User (1 Credit)
DECSYSTEM-20 Assembly Language Programming (2 Credits)
DECSYSTEM-20 Systems Prog (1 Credit)
DECSYSTEM-20 Operation System (2 Credits)

System Administrator
DECSYSTEM-20 User (1 Credit)
DECSYSTEM-20 Systems Admin (1 Credit)

Applications Programmer
DECSYSTEM-20 User (1 Credit)
DECSYSTEM-20 COBOL (1 Credit)

Supplementary Training— Each DECSYSTEM-20 customer's educational needs should be reviewed to determine if additional training (in excess of that provided by the ten training credits) is needed. If so, the additional requirements may be satisfied by the purchase of enrollments in courses conducted in Marlborough or on-site courses.

DECSYSTEM 20 Course Abstracts

DECSYSTEM-20 User
This course is intended for all users of the DECSYSTEM-20 wishing to gain knowledge of the structure of the system and its use from an interactive terminal viewpoint. This course serves as a prerequisite for all other courses in the curriculum.

DECSYSTEM-20 Assembly Language Programming
This course covers the DECSYSTEM-20 instruction set, user level I/O programming, and the MACRO assembler. This course is provided for the system programmer and serves as a prerequisite for the DECSYSTEM-20 Systems Programming Course.

DECSYSTEM-20 Systems Programming
This course covers the more advanced JSYS operations, interfacing with system programs, and the structure of the operating system. This course is provided for the systems programmer and serves as a prerequisite to the Operating System course.

DECSYSTEM-20 Operating System

This course covers the internal working and algorithms of the operating system's Executive and Monitor. The course is intended for the senior systems programmer and should always be preceded by attendance in the Systems Programming and Assembly Language Programming course.

DECSYSTEM-20 Operator

This course is designed to provide the student with the knowledge required for the operation of a DECSYSTEM-20. This course must be preceded by the DECSYSTEM-20 User's Course.

DECSYSTEM-20 Systems Administration

This course is designed to provide the student with the necessary tools for administering a DECSYSTEM-20 installation. The course material is applicable to day-to-day operations, system controls, and available DIGITAL services.

DECSYSTEM-20 COBOL

This course is designed for the COBOL programmer desiring knowledge to write advanced COBOL programs on the DECSYSTEM-20.

DECSYSTEM-20 DBMS

This course is intended for initial users of DBMS including management, systems personnel and applications programmers.

Course Selection Guide

The following guide is provided to aid you in prescribing the right courses for the customer's various personnel.

Personnel Function	Suggested Course Sequence
User Operator	1. DECSYSTEM-20 User (1 Week) 2. DECSYSTEM-20 Operator (1 Week/ on-site)
Systems Programmer (Assembler)	1. DECSYSTEM-20 User (1 Week) 2. DECSYSTEM-20 Assembly Language Programming (2 Weeks) 3. DECSYSTEM-20 Systems Programming (1 Week) 4. DECSYSTEM-20 Operating System
Commercial Programmer (COBOL)	1. DECSYSTEM-20 User (1 Week) 2. DECSYSTEM-20 COBOL (1 Week) 3. DBMS (if applicable) (1 Week)
System Administrator	1. DECSYSTEM-20 User (1 Week) 2. DECSYSTEM-20 System Administration (1 Week)

Software Services

The DIGITAL software organization represents over 3,000 man-years of experience gained from the development and support of operating systems used in more than 50,000 computer installations. DIGITAL's software varies in complexity from real-time executives to minicomputers to the TOPS-10 and TOPS-20. Monitors of the DECsystem-10 and DECSYSTEM-20 applications cover the spectrum from process control and monitoring scientific experiments to implementing reservation and inventory control systems.

DIGITAL offers a wide range of software services. These services range from the personal attention of a skilled software consultant to the distribution of up-to-date software and software information. In this way you can get the most out of your DECSYSTEM-20 and keep pace with advancements in software.

You purchase only what you need. Consulting services are available on a short-term, per-call basis or for a longer-term scheduled or resident period. Likewise, software components, including manuals and updates, can be purchased as part of a software maintenance service or ordered separately from our Software Distribution Center.

Advanced Systems Group

The Advanced Systems Group, an integral part of the DECSYSTEM-20 product group, meets special customer's needs by augmenting the standard products and services of the DECSYSTEM-20. The Advanced Systems Group extends the range of the DECSYSTEM-20 product and services by providing:

Hardware/Software Design and Development—Engineering for special products and modifications to standard products, with emphasis in the areas of communications, network systems, high availability and redundant systems, and special configurations.

Systems Project Management—For large and complex configurations including such functions as systems evaluation and design, systems planning for future expansion, extended factory systems testing, special installation backup and support, and extensions to the DECSYSTEM-20 acceptance test procedures.

Repeat Manufacture—of previously designed and developed special products.

Advanced Systems Group products maintain DIGITAL's high standard of quality. All hardware products are supportable under DECSYSTEM-20 field service support plans and are supplied with full documentation, prints and diagnostics.

Support of special software is provided through a centralized support group on an individual basis for both installation and Software Performance Reports (SPR's).

Because of the customized nature of these products, the Advanced Systems Group provides individual system configuration review, system integration and system installation assistances when needed.

DECSYSTEM 20 Manuals

Documentation is an important part of the DECSYSTEM-20. The planning and production of manuals to support the user with clear, well-written descriptions of the system, programming languages, and command processor was an integral part of the development process.

Manuals are produced using on-line techniques. The text entry and editing capability of DIGITAL's interactive systems result in major user benefits. Because documentation files can be readily updated to reflect improvements made to the software, manuals can be more up-to-date.

Most software documentation is available in complementary quantities from your DIGITAL sales representative.

User Manuals

TITLE	CODE
SYSTEM ERROR DETECTION RECOVERY & REPORTING	EK-SEDRR-RF-001
APL REFERENCE	DEC-20-LARMA-A-D
BASIC-20 KIT DESCRIPTION	LBDDA-
BASIC USER'S GUIDE	LBMAA-
COBOL-20 KIT DESCRIPTION	LCDDA-
COBOL REFERENCE	LCRMA
COBOL-20/SORT-20 KIT DESCRIPTION	LCSDA
COBOL UTILITIES REFER.	LCUMA
FORTRAN-20 KIT DESCRIPTION	LFDDA
FORTRAN REFERENCE	LFRMA
COMMAND LANGUAGE REFERENCE	LLRMA
MACRO ASSEMBLER REFERENCE	LMRMA
SORT-20 KIT DESCRIPTION	LSDDA
BATCH OPERATOR'S GUIDE	OBOGA
BATCH REFERENCE	OBRMA
BEGINNER'S GUIDE TO BATCH	OGBGA
MONITOR CALLS REFERENCE	OMRMA
MONITOR CALLS USER'S GUIDE	OMUGA
TOPS-20 KIT DESCRIPTION	DEC-20-OTPDA-A-D
OPERATOR'S GUIDE	OTPGA
COMPUTER LISTING LABELLED TOPS-20 BWR	OTPRA
USER'S GUIDE	OUGAA
DECSYSTEM 20 USER's GUIDE	PIGAA
ACCOUNTING SYSTEM SPECIFICATIONS	UASSA
EDIT REFERENCE	UERMA
LINK REFERENCE	ULRMA
GETTING STARTED WITH RUNOFF	URGSA
SORT USER'S GUIDE	USTGA
SYSTEM UTILITIES	USYSA
GETTING STARTED WITH DECSYSTEM 20	XGSAA
TOPS-20 SOFTWARE PRODUCT DESCRIPTION	XPDAA
FORTRAN 20 SOFTWARE PROD. DESCRIPTION	XPDAB
COBOL 20 SOFTWARE PROD. DESCRIPTION	XPDAD
BASIC-20 SOFTWARE PRODUCT DESCRIPTION	XPDAE
SORT 20 SOFTWARE PRODUCT DESCRIPTION	XPDAF
SOFTWARE INSTALLATION GUIDE	XSIGA

TERMINAL SESSION

Logging In

To begin a DECSYSTEM-20 session, the user types CONTROL-C (↑C). If available, TOPS-20 responds with the system identification, date and version of the command processor.[1] The user is then prompted with an @ and begins the LOGIN process. [In these examples, the user has used the command recognition and prompting ("novice") modes; TOPS-20 allows the user to type only as much of any command as is necessary for uniqueness.] He then types the escape character, and the command processor determines what command it is and prompts for the next argument. In our example, the colored portions indicate the user's typed input, while the black typeout is produced by the system.

The user types LOG followed by escape and is prompted for a USER.[2] Being a bit uncertain about this, he types a ? and the system defines the field as USER NAME[3] and reprompts up to the last point.[4] The user name DEMO-1 is entered followed by an escape, and the TOPS-20 exec prompts for a password. The password is not echoed to the terminal for security, and the user is then prompted for an account number. The line is terminated by a carriage return. If the login is successful, the user is assigned a Job Number[5] and then prompted for input. The two @ signs indicate the return to command level from a user level command.[6]

```
1 V 1.02.37, 1031 Development Sys., 11-MAR-76. TOPS-20 1A(103)
2 @LOGIN (USER) ?
3 USER NAME
4 @LOGIN (USER) DEMO-1 (PASSWORD)  (ACCOUNT #) 10010
5  JOB 16 ON TTY2 12-MAR-76 14:38
6 @
```

Building a FORTRAN Program

Now the user is ready to begin work. He specifies that he wants to edit a FORTRAN source file called TEST.FOR.[7] Since the file does not yet exist, the editor warns him that a new file is being created,[8] and prompts with line number 00100.[9] The program is now entered line-by-line; each time the editor prompts with a new line number. Input is terminated with an escape (the editor prints it as a $[10] and the user enters command level in the editor, indicated by the asterisk.[11] The programmer notices that he hasn't given A an initial value, and inserts it as line 50. The E command then ends the editing process[12] and the file name is printed as the file is closed.[13]

```
7 @EDIT TEST.FOR

8 %File not found, Creating New file
  Input: TEST.FOR.1
9 00100              DO 5 X=1,1000
  00200              DO 5 J=1,100
  00300    5         A=( (X-1)/(X+1) )%.39*X + A
  00400              WRITE(5,100) A
  00500    100       FORMAT(' A IS ' , E13.6)
  00600              END
10 00700    $
11 *I50
  00050              A=0.
12 *E

13 [TEST.FOR.1]
```

Executing the Program

The user now attempts to execute the program by an EX command (again using prompting) and specifying the file name.[14] TOPS-20 determines from the file extension FOR that this is a FORTRAN program, and calls the compiler automatically.[15] The compiler detects a fatal error in line 300[16] and the LINKing phase, invoked automatically by the EXECUTE command, is bypassed.[17] The user asks for the editor again, and the system assumes, since it wasn't specified otherwise, that he wants to work on the same file he was using the last time he used the editor—TEST.FOR.[18] The user has line 300 printed for reference[19] and then instructs the editor to search for a % and replace it by an * in line 300.[20] The editor retypes the corrected line. The user now leaves the editor with a G command,[21] which causes the new version of the program TEST.FOR.2[22] to be closed (TEST.FOR.1 is still preserved) and specified that the last execution command is to be retrieved (back in).[14]

```
14 @EXECUTE (FROM) TEST.FOR
15 FORTRAN: TEST
16 00300    5        A=( (X-1)/(X+1) )?.39*X + A
   ?FTNIAC LINE:00300 ILLEGAL ASCII CHARACTER % IN SOURCE

   ?FTNFTL    MAIN.          1 FATAL ERRORS AND NO WARNINGS
17 LINK:    Loading
   [LNKNSA No start address]

   EXIT
   @
   @EDIT
18 Edit: TEST.FOR.1
19 *P 300
   00300    5        A=( (X-1)/(X+1) )%.39*X + A
   *
20 *S%$*$300
   00300    5        A=( (X-1)/(X+1) )*.39*X + A
21 *G

22 [TEST.FOR.2]
```

Retry the Program

The FORTRAN compiler is called automatically[23] and this time successfully compiles TEST. The linking loader LINK loads the program and starts execution.[24] Line 400 of the program causes A to be displayed[25] and the program terminates successfully.[26] The user is returned to command level, at which time he inquires note the recognition and prompting about his job.[27] This process can be abbreviated by a user once he becomes familiar with the commands.[28] The user also asks about his quotas of disk storage space,[29] and the system tells him he is allowed 250 working pages (while logged in) and 250 pages of permanent storage (when logged out). The system has a total of almost 18,000 pages available for use.

```
23 FORTRAN: TEST
   MAIN.
24 LINK:    Loading
   [LNKXCT TEST Execution]
25 A IS  0.194420E+08

   END OF EXECUTION
26 CPU TIME: 2.02  ELAPSED TIME: 2.89
   EXIT
   @
27 @INFORMATION (ABOUT) JOB-STATUS
    JOB 16, USER DEMO-1, ACCT 10010, TTY2
28 @I J
    JOB 16, USER DEMO-1, ACCT 10010, TTY2
   @
29 @INFORMATION (ABOUT) DISK-USAGE (OF DIRECTORY)
    6(7) PAGES ASSIGNED, 3 IN USE, 3 DELETED
    3000 WORKING, 2000 PERMANENT ALLOWED
    28500 SYSTEM PAGES FREE
```

Debugging the Program

Although this is a simple example, the user asks to use the interactive FORTRAN debugger to examine the execution of his program.[30] The DEBUG command causes this to happen by recompiling the program, loading it, and starting the debugger,[31] which prompts with a >>. The user now realizes that he needs a clean copy of the program to refer to, and suspends the debugger with a CONTROL-C.[32] He issues a PUSH,[33] which provides him with a separate process (identified by the command processor version number) and then types out his program with a TYPE command.[34] Note that he is not disturbing the suspended debugging session by doing this. The POP[35] command releases this new process and the CONTinue returns to the original task.[36] The user now sets a breakpoint, or pause, at line 300[37] and starts the program running.[38] Before executing that particular line, the FORDDT system regains control and allows the user to type out value of variables in the program. X is typed as a real, J as an integer.[39] When the program is commanded to continue, it stops at line 300 (statement number 5) and waits again.[40] The single command "T"[41] says "type what I last asked for" and we now note that J has changed value. A single "C" causes a continue,[42] and this time the user displays X and A.[43] To change the value for A, he issues an ACCEPT, specifies floating point format, and enters a −1000.[44] Again, a simple continue[45] and the program pauses once more. This time just A is displayed (its value is the new one he entered).[46] He now removes the pause,[47] and continues.[48] The program now runs to completion and exits. Note that the elapsed time for this debugging process was only one minute and thirty-three seconds.

```
  @
30 @DEBUG (FROM) TEST/COMPILE/DEBUG
  FORTRAN: TEST
  MAIN.
  LINK:    Loading
  [LNKDEB FORDDT Execution]

31 STARTING FORTRAN DDT

32 >> ^C
33 @PUSH
    TOPS-20 1A(103)
34 @TYPE TEST.FOR
  00050              A=0.
  00100              DO 5 X=1,1000
  00200              DO 5 J=1,100
  00300    5         A=( (X-1)/(X+1) )*.39*X + A
  00400              WRITE(5,100) A
  00500    100       FORMAT(' A IS ' , E13.6)
  00600              END
  @
35 @POP
36 @CONT


37 >> PAUSE #300

38 >> START

  PAUSE AT 5 IN MAIN PROGRAM

39 >> TYPE X,J/I

  X         =     1.000000

  J         =   1

40 >> CONTINUE


  PAUSE AT 5 IN MAIN PROGRAM

41 >> T

  X         =     1.000000

  J         =   2

42 >> C
```

```
   PAUSE AT 5 IN MAIN PROGRAM

43 >> T X,A

   X         =     1.000000

   A         =    0.0000000

44 >> ACCEPT A/F -1000.

   A         =    -1000.000

45 >> C


   PAUSE AT 5 IN MAIN PROGRAM

46 >> T A

   A         =    -1000.000

47 >> REMOVE

48 >> CON

   A IS  0.194410E+08

   END OF EXECUTION
   CPU TIME: 2.48   ELAPSED TIME: 1:36.30
   EXIT
```

Optimization of the Program

Believing his program to be correct, the user now executes it again, this time asking FORTRAN to optimize the code for faster execution.[49] Comparing the CPU time here to that before,[26] we note the program now runs over three times faster. As a test, the user now wants to direct the terminal output from his WRITE statement to a disk file. To do so, he simply DEFINE's (asking for help with the "?"),[50] the logical device 5 to be on disk (DSK:).[51] Now when the program is executed, we do not see the output on the terminal.[52]

```
49 @EXECUTE TEST/COMPILE/OPT
   FORTRAN: TEST
   MAIN.
   LINK:    Loading
   [LNKXCT TEST Execution]
   A IS  0.194420E+08

   END OF EXECUTION
   CPU TIME: 0.38   ELAPSED TIME: 0.55
   EXIT
   @
50 @DEFINE (LOGICAL NAME) ?
    LOGICAL NAME TO DEFINE OR DELETE,
    OR "*" TO DELETE ALL
```

```
51 @DEFINE (LOGICAL NAME) 5 (AS) DSK:
   @
52 @EX
   FORTRAN: TEST
   MAIN.
   LINK:   Loading
   [LNKXCT TEST Execution]

   END OF EXECUTION
   CPU TIME: 0.45  ELAPSED TIME: 1.13
   EXIT
   @
```

Getting a Directory and Printing a File

To find the file, the user now asks for a directory of his disk area. Terminating the line with a comma[53] puts him in subcommand mode, which is indicated by the double @ @. He now specifies reverse order[54] sort chronologically[55] by creation date.[56] A blank subcommand line terminates that mode and the directory is typed. The newest file is FOR05.DAT.3, the result of a FORTRAN program writing on device 5.[57] The user interrupts the list by CONTROL-C,[58] and asks for the data file to be typed.[59] To get a listing of the file on the line printer, the user issues a LIST command.[60] To see how it will be before the file is printed, he issues a QUEUE inquiry,[61] which shows that his job is being printed now on physical line printer 0. The 24 pages is a maximum limit to prevent accidentally printing excessive worthless pages.

```
53 @DIRECTORY (OF FILES) ,
54 @@REVERSE (SORTING)
55 @@CHRONOLOGICAL (BY) ? ONE OF THE FOLLOWING:
    CREATION
    READ
    WRITE

56 @@CHRONOLOGICAL (BY) CREATION
   @@

      <DEMO-1>
57  FOR05.DAT.1
    TEST.REL.5
      .FOR.2
58    .QOR.1^C
   @
59 @TYPE FOR05.DAT.1
    A IS  0.194420E+08
   @
   @I JOB
    JOB 16, USER DEMO-1, ACCT 10010, TTY2
   @
60 @LIST (FILE) ? FILE NAME
   @LIST (FILE) TEST.FOR
61 @QUEUE
```

```
OUTPUT QUEUE:
DEV          USER                    JOB     SEQ  PRIO  LIMIT   AFTER
PLPTO  *  AARON                    T1003D   441   10     36
LPT       DEMO-1                   TEST     442   10     24
*  Job beins output now

TOTALS:  LPT:    2 Jobs;   60 Pages
```

Finding Out What's Going On

Finally, the user asks for a SYSTAT.[62] Among the information printed is a list of all current users by job, what line they are on, what program they are running (EXEC is the command processor) and their names.[63] Operator service jobs are listed last. In this example, the system has been up for three and a half hours since timesharing began and 26 jobs are in use. Over the last three scheduling periods, the load of jobs in the processor run queue has been 3.8, 2.8, 2.9.

Satisfied with his session, the user logs out,[64] having used 18 seconds of compute time in over 14 minutes of elapsed time.

```
@
62 @SYSTAT
   FRI 12-MAR-76 14:52:24   UP 5:43:01
   16+5 JOBS    LOAD AV    3.8     2.8     2.9

   SYSTEM IS TIMESHARING

63  JOB   LINE  PROGRAM   USER
      2     23   EXEC     MCKIE
      7     15   RUNOFF   KIRSCHEN
      8     63   EDIT     9STUDENT
      9     14   EXEC     MILLER
     10     65   EXEC     3STUDENT
     11     11   EXEC     PORCHER
     12     13   EXEC     HALL
     13      3   EXEC     AARON
     14     26   EXEC     EXERCISER
     15     22   EXEC     HURLEY
     16*     2   EXEC     DEMO-1
     17      4   TECO     FRIES
     19     70   EDIT     11STUDENT
     20     71   EDIT     8STUDENT
     21     62   EXEC     10STUDENT
     23     43   IPRNEW   HEMINGWAY

      1    101   PTYCON   OPERATOR
      3    106   EXEC     OPERATOR
      4    105   OPLEAS   OPERATOR
      5    103   LPTSPL   OPERATOR
      6    104   BATCON   OPERATOR
64 @LOGOUT
   KILLED JOB 16, USER DEMO-1, ACCT 10010, TTY 2, AT 12-MAR-76 14:53:05
     USED 0:0:18 IN 0:14:37
```

☐ Please have a sales representative call me.

☐ Please send me additional information on the DECSYSTEM-20.

NAME_____

COMPANY/INSTITUTION_____

_____

STREET_____

CITY_____

STATE_____ ZIP_____

TELEPHONE_____

My application is_____

_____

_____

_____

_____

_____

_____

_____

_____

063 06076

DECSYSTEM20 Technical Summary

digital

DIGITAL EQUIPMENT CORPORATION
Large Computer Group
Marlborough, Massachusetts 01752
In Europe: Digital Equipment Corporation International, Geneve 26, Switzerland