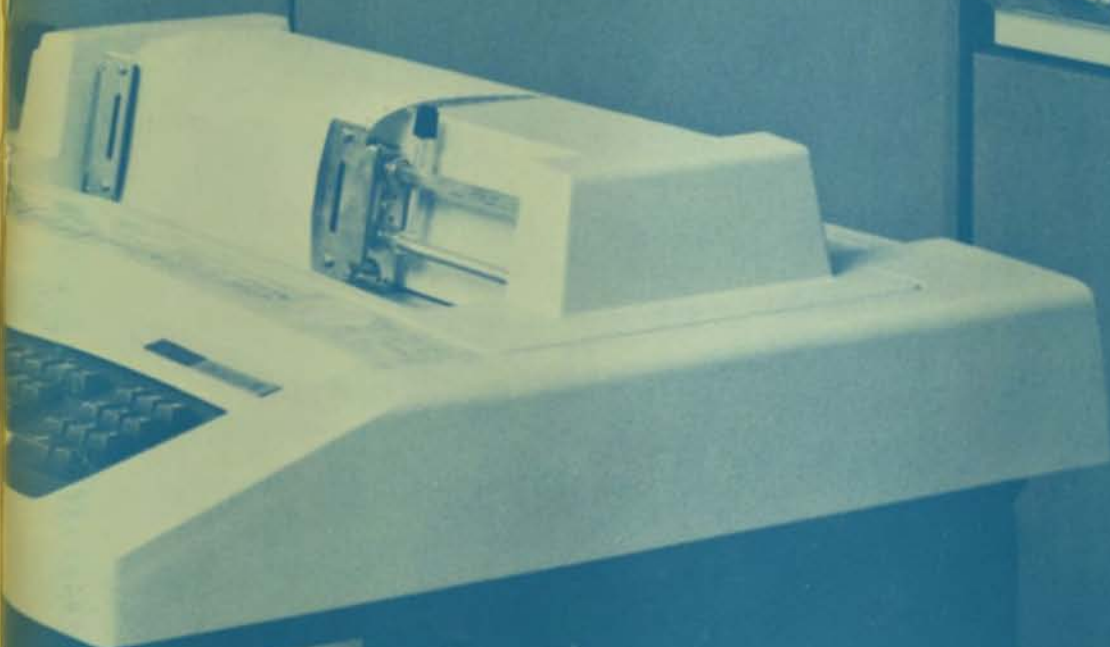


DECSYSTEM-10 TECHNICAL SUMMARY

digital



SYSTEM ARCHITECTURE

The DECsystem-1090	2
Central Processors	4
Instruction Set	6
Instruction Groups	6
Instruction Format	6
Trap Handling on the KL10 and KI10	7
Processor Modes	8
Fast Register Blocks	9
Real-Time Clocks	9
Memory	10
Cache Memory	12
Memory Systems	12
Interleaving	12
Input/Output	13
Priority Interrupt System	13
Multiplexed I/O Bus	13
Massbus	14
I/O Channel Capacity	14
Console/Diagnostic Computer	15
System Integrity Features	16
Power Fail	16
Error Handling	16
Multiprocessor Systems	17
Dual CPU System	17

PERIPHERALS

PERIPHERALS	19
Disk Systems	20
DAS33 Buffered Data Channel	22
DECtape Transport	23
Magnetic Tape Systems	24
Hard Copy Equipment	26
Card Readers	26
Line Printers	26
Terminals	28

DATA COMMUNICATIONS

DATA COMMUNICATIONS	33
System Concepts	34
Asynchronous Communications	34
Synchronous Communications	34
Synchronous vs. Asynchronous Transmission	35
Remote Communications	36
Network Concepts	37
Simple Topologies	37
Complex Topologies	37
Task-to-Task Communication	38
IBM Device Emulation and Termination	38
Communications Products	39
DN87/87S Universal Synchronous/Asynchronous Front End Subsystem	39
Complex Topologies	42
DECnet-10	42
DAS80 Series Remote Station	43
DAS92 Remote Station	45
DAS 61 IBM 2780/3780 Support	46
DAS 62 IBM HASP Multi-leaving Support	46

CONFIGURATOR

CONFIGURATOR	47
--------------	----

SOFTWARE

SOFTWARE	59
Languages	61
AID	60
ALGOL	60
APL	60
BASIC	62
COBOL	61
ISAM	62
CPL	62
FORTRAN	63
FORTRAN Version 4	63
MACRO	64
System Utilities	65
SORT	65
Data Base Management System	65
ITPS-10	66
LINK-10	66
MACY11-LNKX11	67
DDT	67
FAILSAFE/BACKUP	67
Message Control System	68
PIP	68
RUNOFF	68
TECO	68
Operating System	69
TOPS-10 Operating System	69
Command Control Language	72
File System	73
Input/Output	76
Memory Management	76
Multi-Processor Systems	77
Inter-job Communication	78
Communications Software	78
Multimode Computing	79
Timesharing	79
GALAXY-10 Batch	79
Real-Time Computing	81
Diagnostic Software	83
High Availability	83
Maintenance Features	83
On-Line Diagnosis	83

SERVICES

SERVICES	85
People	87
Parts	87
Support	87
Service Agreements	88
Education	88
Logistics	89
Technical Documentation	89
Software Services	90
Advanced Systems Group	91

THE DECsystem-1090

The DECsystem-1090 is an extension of the DECsystem-1080 which was announced in September of 1974 and first delivered in June of 1975. The DECsystem-1090 offers all the outstanding features of the 1080 plus many new ones. The key features of the DECsystem-1090 are:

The new KL10B Processor.

It features 1.4 mips, integrated channels, cache memory and a PDP-11 Console front end computer which permits on-line, remote diagnosis and maintenance. The new CPU permits more compact configuration and lower power consumption through the use of integrated channels; improved performance; and substantially lower system purchase and maintenance costs. Single and dual CPU systems, featuring high availability, are offered.

The new MH10 Memory System.

It offers 256K 36-bit words in one 31" system cabinet. Internal 2-way interleaving and eight memory ports accommodate up to 2 CPU's and all of DIGITAL's standard memory channel peripheral subsystems. The DECsystem-10 memory can be expanded with up to 16 MH10 units to over 4 million words or 20 million ASCII characters.

The new DN87S Communications front end.

It provides additional features and lower costs than previous communication front end offerings. Cost is reduced through the use of the integrated DTE channels which utilize high-speed microcode for data transfer. Important new features include DECnet compatibility and simultaneous support of synchronous and asynchronous communications on a single front end. All the standard ANF features are also supported. Up to three DN87S units per 1090 System can accommodate up to 336 full duplex asynchronous lines or 36 synchronous lines. The number of lines can be extended by adding a DN87A and up to three DN87D's, each of which can support 112 asynchronous or 12 synchronous lines.

The recently-announced RP06 200 megabyte Disks.

They are now available using the integrated MASSBUS® controllers. Not only is the cost of Disk Systems substantially reduced, but the use of integrated controllers and channels provide automatic update of the cache memory—thus reducing overhead. The on-line disk data storage can now be extended to over 6.2 billion ASCII characters or about 5 billion 8-bit bytes.

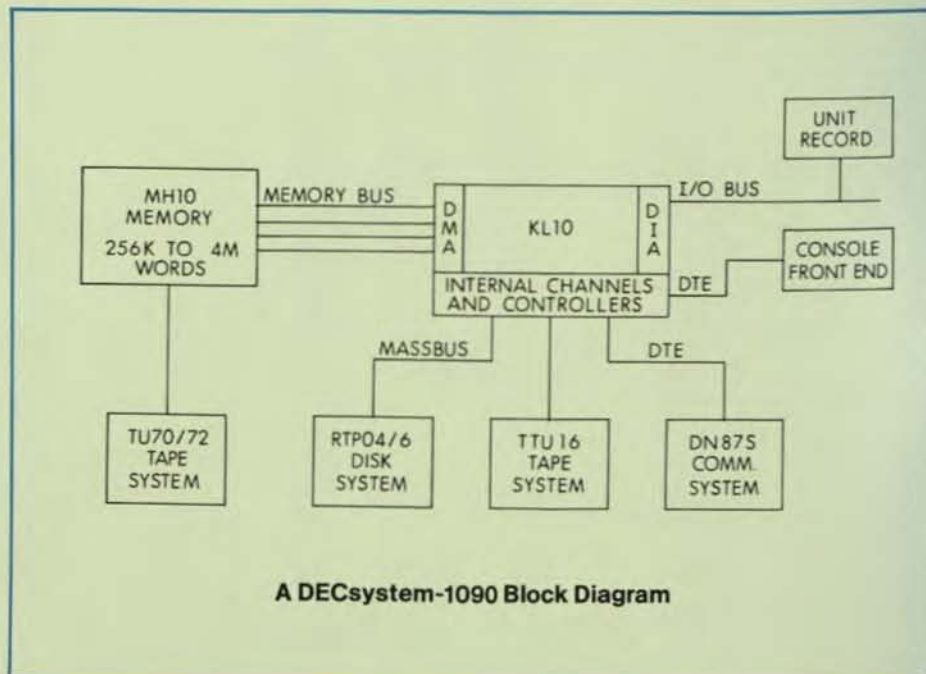
The new TTU16 Magnetic Tape Subsystem.

It now uses the integrated MASSBUS® Controllers to extend the DECsystem-1090 tape capability to three subsystems. Transfer rates of 72K (TU16),

320K (TU70) and 780K (TU72) characters per second allow DECsystem-1090 users to select the tape system which meets their needs. With the new TU72 system, 6250 bpi capability has been added to our product offerings at the high end and the TTU16 72KHz system greatly reduces the cost of 1600 bpi capability, for low performance requirements.

A complete range of Line Printers and Card Readers.

The new LP100 offers high quality "band" printing at 1200 to 1500 lines per minute. The LP10 and LU10 offer high quality mid-range and low-speed printers at lower cost to meet users' individual needs. The CR10E and CR10F 1200 and 300 card per minute readers offer a choice of two excellent card readers.



SYSTEM ARCHITECTURE

Central Processors	4
Instruction Set	6
Instruction Groups	6
Instruction Format	6
Trap Handling on the KL10 and KI10	7
Processor Modes	8
Fast Register Blocks	9
Real-Time Clocks	9
Memory	10
Cache Memory	12
Memory Systems	12
Interleaving	12
Input/Output	13
Priority Interrupt System	13
Multiplexed I/O Bus	13
Massbus	14
I/O Channel Capacity	14
Console/Diagnostic Computer	15
System Integrity Features	16
Power Fail	16
Error Handling	16
Multiprocessor Systems	17
Dual CPU System	17

Central Processors

Within the family of six DECsystem-10 configurations, there are three central processor units. All operate under the same DECsystem-10 operating system (TOPS-10), execute the same software, and share most models of DECsystem-10 peripheral equipment. The three processors differ in their speed, method of address mapping, memory capacity, program size, auto-diagnostic and restart features, and priority interrupt system.

The KA10 central processor is used in the DECsystem-1050 and 1055. The more powerful KI10 central processor is used in the DECsystem-1070 and 1077 configurations. The top-of-the-line KL10 central processor is used in the DECsystem-1080, 1090 and 1099. The high speed 2K word cache memory (K=1,024) permits rapid instruction execution while minimizing accesses to main memory. A double precision floating multiply, for instance, executes in 4.8 μ sec.

Because all processors operate with the same operating system, it is easy to field upgrade a system with a KA10 and replace it with a KI10 processor; or take a system with a KI10 and replace it with a KL10 processor, providing a significant increase in the computing power with no necessary changes in user programs.

The 1090 configuration differs from the 1050 or 1070 in that the high speed data channels and communications processor channels are integral to the central processor. Up to eight high speed buffered data channels can be utilized which support DIGITAL's Massbus® peripherals. Because the data channels access four 36-bit words of memory in one channel request cycle the total throughput is in excess of one million words per second or 4 megabytes per second. For non-Massbus® peripherals, external channels which transfer directly to memory can be utilized. An important feature of the integrated high speed data channels is that cache memory is automatically updated

during I/O transfers. The integrated front end processor channels will accommodate up to three PDP-11 communications front ends in addition to the KL10 console front end. These channels utilize high speed microcode to transfer and process data coming from the PDP-11 communications front ends. All channel data transfers are parity checked. A final benefit of the 1090 internal channels is configuration simplicity, improved maintainability and reduced power consumption.

DECsystem-10 at a Glance.

	1040/1050	1055*	1060/1070	1077*	1080/1090	1088*/1099*
Relative Performance	1.0	1.8	2.2	3.5	4.8	6.7
Avg. Number of Users	10-40	20-70	30-80	40-100	50-127	75-150
No. of CPUs	1	2	1	2	1	2
Memory Size in K words (min.-max.) (K=1024)	64-256	128-256	128-4096	128-4096	128-4096	128-4096
No. of Instructions:	366	366	378	378	398	398
Instruction Look-ahead	No	No	Yes	Yes	Yes	Yes
Virtual Memory	No	No	Yes	Yes	Yes	Yes
Memory Interleaving	2 or 4 way	2 or 4 way	2 or 4 way	2 or 4 way	2 or 4 way	2 or 4 way
Index Registers	15	15 each CPU	4 x 15	4 x 15 each CPU	8 x 15	8 x 15 each CPU
Accumulators	16	16 each CPU	4 x 16	4 x 16 each CPU	8 x 16	8 x 16 each CPU
Instruction Times (microseconds)**						
Fixed Point Add	2.8	2.8	1.6	1.6	.52	.52
Fixed Point Multiply	9.8	9.8	4.6	4.6	2.4	2.4
Unconditional Jump	1.5	1.5	1.1	1.1	.36	.36
Single Precision Floating Point Add	9.8	9.8	2.7	2.7	1.8	1.8
Double Precision Floating Point Add	59.4	59.4	3.9	3.9	2.2	2.2
Double Precision Floating Multiply	N.A.	N.A.	8.1	8.1	4.8	4.8
Double Precision Floating Divide	N.A.	N.A.	16.0	16.0	10.2	10.2
Increment and Move Byte	N.A.	N.A.	5.6	5.6	1.4	1.4
Move from Memory	N.A.	N.A.	1.6	1.6	.48	.48
I/O Bus Band width (words/second)	200K	200K	370K	370K	370K	370K
Memory Bus Band width (words/second)	4000K	4000K	4000K	4000K	4000K	4000K

* Dual-processor systems execute two instructions simultaneously.

**For 1080/1088 and 1090/1099 Operands in Cache Memory.

Instruction Set

Instruction Groups

The KA10 has 366 instructions, the KI10 378 instructions, and the KL10 398 instructions, an extremely large repertoire which provides the flexibility required for specialized computing problems. The instruction sets of the KA10 and KI10 are hardwired. By contrast, the instruction set of the KL10 is micro-programmed; that is, each instruction is actually a series of micro-instructions that perform various logical functions such as processor state control, data path control and actual implementation of each instruction in the KL10's set. The micro-code is loaded into a 1K 80-bit word RAM (random access memory) through the PDP-11 console/diagnostic computer. Since the set provides so many instructions to choose from, fewer instructions are required to perform a given function. Assembly language programs are therefore shorter than with other computers, and the instruction set simplifies the Monitor, language processors, and utility programs. For example, compiled programs on a DECsystem-10 are often 30 to 50 percent shorter, require less memory and execute faster than those of comparable computers.

In addition to these instructions, the DECsystem-10 provides 64 programmable operators, 33 of which "trap" to the Monitor (Monitor calls) and 31 of which trap to the user's call area. The remaining instruction codes are unimplemented and reserved for future expansion. An attempt to execute one of these unimplemented instructions results in a trap to the Monitor.

The instruction set, despite its size, is easy to learn. It is logically grouped into families of instructions and the mnemonic code is constructed modularly. All instructions are capable of directly addressing a full 256K (36 bit) words of memory without resorting to base registers, displacement addressing, or indirect addressing. Instructions may, however, use indirect addressing with 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 speed execution.

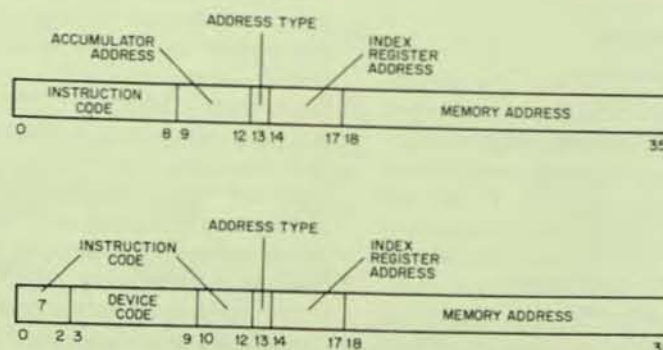
Instruction Format

In all the non-input/output instructions, the nine high-order bits (0-8) specify the operation, and 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 always 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 a memory address.

The instruction codes that are not assigned as specific instructions are performed by the processor as so-called "unimplemented operations", as are the codes for floating point and byte manipulation in any KA10 that does not have the hardware for these instructions.

An input/output instruction is designated by three 1s in bits 0-2. Bits 3-9 address the input/output device to be used in executing the instruction, and bits 10-12 specify the operation. The rest of the word is the same as in other instructions.



INSTRUCTION FORMAT

HALF-WORD DATA TRANSMISSION

The half-word data transmission instructions move a half-word and may modify the contents of the other half of the destination location. There are 16 instructions which differ in the direction that they move the chosen half-word and in the manner in which they modify the other half of the destination location.

FULL-WORD DATA TRANSMISSION

The full-word data transmission instructions move one or more full words of data from one place to another. The instructions may perform minor arithmetic operations such as forming the negative or the magnitude of the word being processed.

BYTE MANIPULATION

The five byte manipulation instructions pack or unpack bytes of any length anywhere within a word.

LOGIC INSTRUCTIONS

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

FIXED-POINT ARITHMETIC

Two common conventions are to regard 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} or -1 to $1-2^{-35}$. Since multiplication and division make use of double-precision numbers, there are special instructions for performing these operations to yield results that can be represented by a single word.

The format for double-length fixed-point numbers is just an extension of the single-precision 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, 0 for positive, 1 for negative. 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 $1-2^{-70}$. Zero is represented by a word containing all 0s.

FLOATING-POINT ARITHMETIC

All DECsystem-10 processors have instructions to perform scaling, negating, addition, subtraction, multiplication and division upon numbers in single-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.

As a superset of the KA10, the KI10 and KL10 have instructions to perform all of the above functions in double-precision floating-point format as well as single-precision floating-point. In double-precision floating-point format, one bit is used for the sign, eight bits are used for the exponent and 62 bits are used for the fraction.

FIXED/FLOATING CONVERSIONS

Special KI10 and KL10 instructions provide the capability of converting fixed-point formats to or from floating-point formats. Two sets of instructions are provided to perform this function: one set optimized for FORTRAN and a second set optimized for ALGOL.

ARITHMETIC TESTING

The arithmetic testing instructions may jump or skip, depending on the result of an arithmetic test and may first perform an arithmetic operation on the test word.

LOGICAL TESTING, MODIFICATION, AND SKIP

These instructions modify and/or test using a mask and/or skip on selected bits in an accumulator.

PROGRAM CONTROL

Program control instructions include several types of jump instructions and the subroutine control PUSHJ and POPJ instructions.

INPUT/OUTPUT OPERATIONS

Input/output instructions govern all direct transfers of data to and from the peripheral equipment and also perform many operations within the processor. Block transfer instructions handle bulk data transfers to and from I/O devices.

UNIMPLEMENTED USER OPERATIONS (UOUs)

Many of the codes not assigned as specific instructions are executed as unimplemented user operations wherein the word given as an instruction is trapped and must be interpreted by a routine included for this purpose either by the programmer or by the Monitor. Those UOUs reserved for use by the Monitor are called Monitor UOUs (MUOUs), while user UOUs are called Local UOUs (LUOUs). Instructions that are illegal in user mode also trap in the same manner as MUOUs.

BUSINESS INSTRUCTION SET

Five instructions comprise the Business Instruction Set in the KL10 central processor. Four of these are new arithmetic instructions to add, subtract, multiply, and divide using double precision, fixed-point operands. The new STRING instruction is capable of performing nine separate functions.

These functions include an EDIT capability; decimal to binary and binary to decimal conversion in both offset and translated mode, (offset mode is byte modification by addition of 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 ACs and can detect special characters in the source string.); Move String in both offset and translated mode; and Compare String in both offset and translated mode.

This Business Instruction Set provides faster processing since there are specific instructions for doing more comprehensive string operations. These instructions can be used on a variety of code types such as ASCII and EBCDIC.

Trap Handling on the KI10 and KL10

The KI10 and KL10 provide facilities for handling arithmetic overflow and underflow conditions, pushdown list overflow conditions, and page failures directly by the execution of programmed trap instructions. This trap capability avoids recourse to the program interrupt system.

If a program is running in a public submode, pages within the user's addressing space are accessible only if they are listed in the user's page map and are defined to be accessible from public mode. Pages designated public are, by definition, accessible. Pages designated concealed may be accessed only at defined entry points, i.e. portals which permit entry from public submode programs. In concealed submode operations, programs can access all of the virtual addressing space. However, if a program running in concealed submode executes an instruction from an area designated to be public, the state of the processor transfers over into public submode. Concealed areas can be used for proprietary coding that can be executed but not altered or examined by users in the public mode.

The supervisor and kernel submodes are similar but not identical to the public and concealed submodes. Supervisor submode programs can access but

cannot alter areas designated as concealed. Also, any instruction executed out of a public area from either supervisor or kernel submode returns the processor to supervisor submode. In kernel submode operations all of memory is accessible and can be altered. Programs operating in kernel submode can address portions of memory directly, without paging, and it is through the kernel submode program that page restrictions are established. Functions delegated to supervisor submode generally include those affecting individual users as opposed to overall system management of input/output, priority interrupts, page map accounting, etc., which are handled by kernel submode programs. The ability of kernel submode programs to supply information which supervisor submode programs can read but not alter allows portions of the operating system to be hardware-protected from other portions undergoing modifications or design changes.

Processor Modes

Instructions on the KA10 are executed in one of two modes depending upon whether a mode bit has been set. Programs operate in either User Mode or Executive Mode. In Executive Mode operations, all implemented instructions are legal, addresses are not relocated, and all core locations are accessible. The Monitor operates in Executive Mode and is able to control all system resources and the state of the processor. In User Mode operations, addresses are relocated, certain instructions are illegal, causing monitor traps when executed, and address references are confined within two user program segments.

The KI10 and KL10 further divide Executive and User Mode operation into two submodes each. User Mode is subdivided into public and concealed submodes and Executive Mode into supervisor and kernel submodes. For each 512-word page in the system, information is stored in a table maintained by the operating system which specifies whether or not a page can be accessed or altered, and if it is defined to be public or concealed.

KI10 AND KL10 PROCESSOR MODES

USER MODE

Public Submode

- User programs
- 256K word address
- All instructions permitted unless they compromise integrity of system or other users
- Can transfer to concealed submode only at entry points

Concealed Submode

- Proprietary programs
- Can READ, WRITE, EXECUTE or TRANSFER to any location labeled Public

EXECUTIVE MODE (Monitor)

Supervisor Submode

- Performs general management of system
- Performs those functions that affect only one user at a time
- Executes in virtual address space labeled public

Kernel Submode

- Performs I/O for system
- Performs those functions that affect all users

Fast Register Blocks

General-purpose registers are a DEC-system-10 feature that help improve program execution. These sets of fast integrated circuit registers can be used as accumulators, index registers, and as memory locations. Since they may be addressed as memory locations, they do not require special instructions.

One set of 16 registers is included in the KA10, four sets of 16 fast registers are included in the KI10, and eight sets of 16 fast registers are included in the KL10. Context switching on the KA10 is performed by storing the register information into core locations. Program switching time for the KL10 between register stacks is 500 nanoseconds.

On the KI10 and KL10, different register blocks can be used for the operating system and individual users. This eliminates the need for storing register contents when switching from User Mode to Executive Mode. Also, a critical real-time program is able to maintain its own register block for handling data and interrupt sequences at maximum speed.

Real-time Clocks

The DK10 Real-time Clock is available with each KA10- or KI10-based DEC-system-10 and provides high-resolution timekeeping for time accounting, time-base maintenance, periodic high-frequency interrupts, and interval timing. Meeting the most demanding real-time requirements, the clock provides 10 μ sec resolution and a choice of up to 2^{18} possible timing intervals, so that interrupts can be programmed at intervals from 10 μ sec up to 2.6 seconds.

In addition to an interval register, the DK10 has a frequency counter which counts the pulses of an internal 100 kHz ± 0.01 clock, or an external clock having a maximum frequency of 400 kHz. The clock includes a comparator network which provides a running comparison between the frequency counter and the internal register. When the frequency counter reading equals the total on the internal register, a program interrupt is generated and the frequency counter is automatically reset so that it can time the next interval.

The clock, which is assignable to any interrupt channel, can be used to pace real-time, monitor, or other functions performed in either Executive or User Modes. In fact, a KA10 or KI10 system can have two clocks—one for each mode—since two device codes are available for clock use. Clock updating is interlocked with the DATAI instructions so that it can be read correctly—at any time—by the KA10 or KI10 without losing a clock pulse.

The four clocks built into the KL10 provide a number of timing and counting functions including an interval timer, a time base, an accounting meter and a performance analysis counter.

All of the operations on these clocks are accomplished by means of I/O instructions to internal devices. Many of these functions use a microsecond source of pulses which is counted down from the basic machine clock. The 0.005% tolerance will give less than five seconds drift over 24 hours.

The Interval Timer provides a programmable source of interrupts with a 1 microsecond resolution and is similar to the DK10 real-time clock. It is used for real-time applications and for page management by the Monitor. It is designed so that a real-time deadline schedule with varying deadlines can be implemented.

The Time Base provides a 60-bit microsecond resolution source of elapsed time. This gives over a 9,140 year maximum time before wrap-around.

The Accounting Meters provide an accurate and reproducible measure of the amount of processor resource used by a job, interrupts, page failures, and cache activity.

The Performance Analysis Counter provides a tool for studying hardware and software performance of the system. It may be used to point to hardware and/or software bottlenecks.

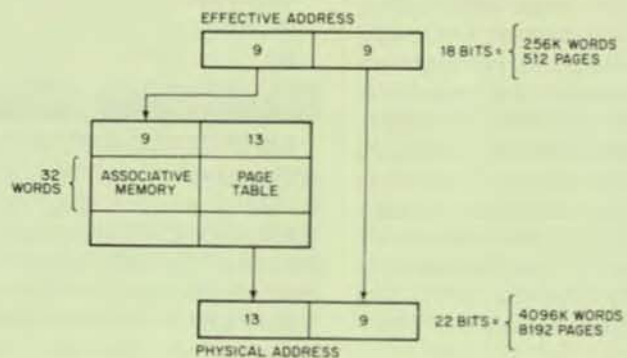
Memory

On the KA10, dual memory protection and relocation registers protect other users and the monitor from errors in the active user program. If a user attempts to reference memory outside his area, his program is stopped from operating and control is transmitted to the operating system. The feature also permits dynamic memory allocation, allowing each user program to be assigned either one or two segments of memory. One of these segments can be write-protected so as to protect critical programs or data from errors committed by the user. Segments—which are multiples or 1024 words—may be located anywhere in memory and may contract and expand as needs dictate. Memory management techniques, employed by the Monitor, insure efficient memory usage.

The division of programs into two segments also makes possible the reentrant or recursive use of language processors and systems programs. With the DEC-system-10, all DIGITAL-supported language processors and most utility programs are reentrant. In other words, each program is written in two parts. One part contains pure or reentrant code that is not modified during execution so the same copy can be used to simultaneously service any number of users. A separate second part of the program belongs strictly to each user and consists of non-reentrant code and data. This section is stored in a separate area of core. This feature allows more users to utilize a given amount of physical core simultaneously. Alternatively, less core is required to service a given number of users. The result is much more efficient use of computing resources.

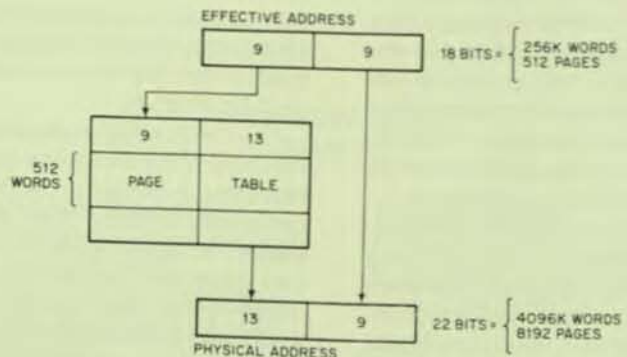
The ability to divide programs into two segments is not limited to DIGITAL-supplied language processors and utilities. User-written programs, including those written in higher-level languages such as FORTRAN and COBOL, can also take advantage of this feature. Thus, user developed applications can benefit from the same efficient treatment given DIGITAL-supplied software.

KA10 Processor Associative Memory



KA10 ASSOCIATIVE MEMORY

KL10 Processor Page Addressing



KL10 PAGE ADDRESSING

The KI10 and KL10 provide memory address mapping from the program's memory address space (referred to as the effective address) to the physical memory address space by substitution of the most significant bits of the memory address. This mapping provides access to the entire physical memory space which can be up to 16 times larger than the maximum user address space. The user's effective address space is 256K words addressed with 18-bit addresses; the physical address space is 4,096K words addressed with 22-bit addresses (where 4,096K is equivalent to 4,194,304 decimal).

The memory mapping process utilizes the most significant 9 bits of the effective address as an index into the appropriate page map (User or Executive). The data located by the index provides 13 bits which are appended to the least-significant 9 bits of the effective address in order to form the 22-bit physical address. Also provided are 3 bits which indicate what type of memory requests are allowed to the page in question (none, read-only, proprietary, etc.).

If this scheme were implemented exactly as outlined above, every user memory reference would require two actual memory references: one to obtain the memory mapping data and one to obtain the user's mapped memory reference. In order to reduce the number of actual memory references to nearly the same number as required by the program, an associative memory mapping unit is used in the KI10, while a copy of the entire page table is kept in the KL10 hardware.

If the address is in the range 0-17 (octal) inclusive, the hardware fast register blocks are referenced instead of the memory system. On the KI10, the User Mode bit and the high-order 9-bits of the virtual address are compared against the contents of the associative memory registers which are part of the memory mapping hardware. These 10-bits will either exactly match one of the associative registers or a no-match occurs. On the KL10, the User Mode bit and the high-order 9-bits of the virtual address are used to do a "table look-up" in the hardware page table.

On the KI10, if a match exists, the contents of the related register supply the 13-bit most-significant portion of the physical memory address and also supply 3 bits which indicate what type of memory references are allowed to this page, i.e., if the memory request is not consistent with the request type allowed bits, a page failure occurs. On the KL10, if a page table entry exists, the contents of the entry supply the 13-bit most-significant portion of the physical memory address and also supply 3 bits which indicate what types of memory references are allowed to this page.

For both the KI10 and KL10, if the memory request is consistent with the request type allowed bits, the physical address used consists of the 13 bits from the related register as the most significant bits of the physical address and the 9 least significant bits of the effective address as the least significant bits of the physical address.

When the relocation data for a referenced page does not exist in the associative memory of the KI10, or the hardware page table of the KL10 (i.e., a no-match), the hardware reads the relocation data from the page table in memory and stores it into the associative memory of the KI10 or the hardware page table of the KL10.

The associative memory in the KI10 has associated with it a reload counter that points to one of the 32 entries in the memory.

Any time a word of the associative memory of the KI10 is referenced and the reload counter is pointing at it, the reload counter is incremented to point at the next word in the KI10 associative memory. Whenever it is necessary to load a new entry into the associative memory, the location replaced is the one to which the reload counter points. Thus, at worst, a page which was just referenced would not have its word in the KI10 associative memory replaced by the next memory reference. If a particular word in the KI10 associative memory has not been referenced in some time, the reload counter would be left pointing to this word, having been pushed away from all words in associative memory which have been used. Thus, a 1-bit approximation to "least-recently-used" page table operation is obtained.

The monitor assigns the core area for each user by loading the various page tables, setting up the trap locations in the user page map, and responding appropriately when a trap occurs. The Monitor provides memory protection for itself and each user by filling the page tables only with those entries which are allowed to be accessed. A zero access bit in the page table will cause a reference to the associated page to initiate a page failure trap to the Monitor.

The TOPS-10 Operating System utilizes the KI10 and KL10 in-core page maps to create one or two segment programs in roughly the same fashion as it uses the protection and relocation registers of the KA10. The major benefits of the paging capability are a smaller unit of core allocation (512 words instead of 1024), the freedom to scatter the pages of a segment randomly through physical core (avoiding core fragmentation and the overhead of repacking core), and the opportunity to execute a program when all of its pages are not in physical core (i.e., a virtual memory capability).

Memory Systems

To meet the requirements of large systems, DECsystem-10 core memory can be modularly expanded to 256K words for KA10-based systems or 4,096K words for KI10- and KL10-based systems, all directly addressable. Memory can consist of combinations in 64K, 128K and 256K word modules. The structure of the memory bus gives the central processor and high-speed data channels simultaneous access to separate memory modules and allows each to operate at its own speed.

The MF10 provides up to 4 ports and the MG10 or MH10 up to 8 ports. Each port can be further expanded through the use of the MX10 memory multiplexer. The multiplexer handles up to eight channels, inter-leaving data from the channels on a word-by-word priority basis. Such parallel operation yields many improvements over systems which provide only a single path to memory. The memory bus station allows each data channel to transmit full 36-bit words in parallel at a speed of one million words per second. In total, the memory structure operates at rates of up to 10 million characters per second when I/O devices and processors are concurrently simultaneously transferring data. In addition, the memory bus is capable of handling a maximum of 16 memory modules. Each memory module provides switches which allow that particular module to represent any module of its size in the addressable memory space. Thus, one model can replace another without rewiring. The Switches also provide for memory interleaving.

The core memory systems available for the DECsystem-10 are described in the memory table below.

Interleaving

The concept of interleaving has long been used to increase the effective bandwidth of DECsystem-10 systems. On KA- and KI-based systems, successive memory locations were located, via memory addressing hardware, in alternating memory modules. Thus, the processor, when requesting successive logical locations, would not have to wait for the complete memory read/write cycle to be complete before affecting a fetch. The first memory module could be completing the write cycle, while the second memory module could be accomplishing a read cycle.

The memory implementation techniques for the KL dramatically extend this concept. The memory bus architecture of the KL10 differs from both the KA10 and the KI10 in that its width is four 36-bit words. That is, in an adequately configured memory system, the KL10 actually reads or writes four words concurrently from memory. The four logically successive memory locations (obtained from four physically separate memory modules) are placed in the KL's cache memory, and in typical programs, another core memory reference need not be initiated until the four instructions or data have been processed. Combined with the typical effects of the cache memory, therefore the KL has an effective memory word access time of 334 nanoseconds, using current DIGITAL memory.

Cache Memory

The KL10 features a cache or buffer memory with 160 nanosecond access time. Data being written to or read from memory is typically found in the cache 85 to 90 percent of the time, thus giving the KL10 an effective memory access time of 334 nanoseconds with current DECsystem-10 memories. Another feature of the "state-of-the-art" design cache memory on the KL10 is that 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.

A cache sweep feature allows all pages or one selected physical page in main memory to be updated from the cache.

Core Memories

	MH 10	MG10	MF10
Word Size	36 bits plus parity	36 bits plus parity	36 bits plus parity
Minimum Memory Size	64K	32K	32K
Maximum Memory Size			
KA10	256K	256K	256K
KI10	4096K	2048K	1024K
KL10	4096K	2048K	1024K
Module Sizes			
32K words	No	Yes	Yes
64K words	Yes	Yes	Yes
128K words	Yes	Yes	No
256K words	Yes	No	No
Read Access Time			
Typical	735	620 ns	550 ns
Maximum	745	660 ns	610 ns
Cycle Time	1.2 μ sec	1.0 μ s	950 ns
Interleaving	2 or 4 way	2 or 4 way	2 or 4 way
Minimum memory that can be disabled	64K	32K	External Only 32K

Input/Output

Priority Interrupt System

The DECsystem-10 Priority Interrupt System is one of the most flexible available today. 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. Each interrupt level has any number of high-speed programmable sublevels. Thus, 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, all or any combination of levels with a single instruction. In addition, the program can assign some or all devices to the same level.

A set of instructions (block-in and block-out) allow blocks of information to be transferred between a device and memory with a single instruction. These instructions identify the source of the interrupt, update a word count and data address, transmit or receive the block or information, and dismiss the interrupt.

The system can also generate interrupts through software. Real-time hardware can thus operate on a high-priority level while related computations, particularly if they are lengthy, can be performed on a lower level.

The DECsystem-10 program-assignable priority interrupt system provides much greater flexibility than permanently hard-wired systems. Hard-wired systems require a large number of levels, often operate with an extremely high overhead, and cannot change device priorities without system shutdown and rewiring.

An interrupt on the KI10 or KL10 can cause the processor and the interrupting device to immediately initiate one of several possible actions. In response to the "interrupt grant" signal from the processor, 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 (KI10 or KL10)
- Transfers a word into or out of the addressed location (KI10 or KL10)
- Adds a signed 11-bit (12 bits+ + 11 bits) value to the addressed location (KL10 only)
- Traps to a predefined memory location (KA10)

Peripheral devices which are not equipped with the decoding logic perform an interrupt and transfer of control, as on the KA10, to one of the standard interrupt trap locations.

Multiplexed I/O Bus

The DECsystem-10 Multiplexed I/O Bus provides a 36-bit full word parallel path between memory and an I/O device for purposes of control or low-speed data transfer. To initiate high speed data transmission directly between memory and a device connected to the memory bus, a control word is first transferred over the I/O bus to the buffer of the high-speed device controller. Then, on command, entire data blocks are moved directly to or from memory with a single instruction. (For a description of the memory bus, see the discussion of Memory Systems.)

The I/O bus may also be used as a control and data path to/from a large number of low-speed I/O devices. Transfer is performed in 36-bit words in parallel at speeds of 200K words/sec on the KA10, and 370K words/sec on the KI10 and KL10. Thus each data transmission instruction moves one word of data between memory and the buffer of the device controller. When block input or output instructions are used, entire blocks of data are moved to or from the device with a single instruction.

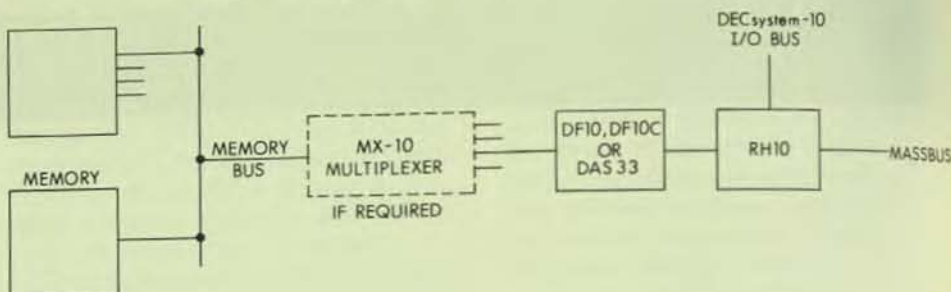
Massbus

The KL10 architecture and implementation now allow use of DIGITAL's latest Massbus controllers, channels, and peripheral devices. Although, via memory bus and I/O bus logic, the KL10 supports traditional peripheral control units, the efficient memory control unit provides paths and logic for transfers into and out of DECsystem-1090 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, there is one RH20 channel containing a 16-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.

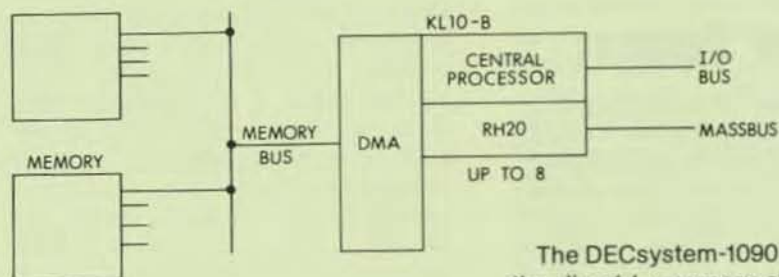
A storage bus provides a 1.6 million words/second transfer bandwidth between the memory control unit and memory. In diagnostic mode, the storage bus is used to access the internal memory controllers. The channel bus is a physically short, high-speed data transfer path providing a peak I/O bandwidth of 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.

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.

1040, 1070, 1080 MASSBUS CHANNEL AND CONTROL BLOCK DIAGRAM



1090 MASSBUS CHANNEL



I/O CHANNEL CAPACITY

DECsystem-1050 and 1070 generation equipment implement high-speed data transfers from mass storage devices via a direct to memory channel using the DF10, DAS33 or DX10 channel and an appropriate controller. If the number of channels exceeds the available memory ports then an MX10 multiplexer can be used. In general, capacity is determined by the memory system word transfer rate capacity which will vary based on the type of memory used and the method of interleave. For current memories, four words can be transferred every microsecond in a four-way interleaved configuration. Conflict in memory port contention will reduce the effective word transfer rate somewhat and can result in "overruns" which occur when a rotating mass storage device cannot execute a memory transfer in the required time. TOPS-10 recovers from this situation and simply retries the transfer until it is completed. The DAS33 buffered channel will greatly reduce or eliminate "overrun" capacity loss.

The DECsystem-1090 can use either the direct to memory channels or the internal channels in the KL10B Central Processor. Since the processor is configured on a lower memory priority port than direct to memory channels there is no conflict between these devices. The KL10 cache and internal buffering further reduce any contention. The number of internal high-speed channels which can be operated simultaneously depends on the data transfer capacity between the CPU and Memory. This, in turn, is a function of the memory interleave method and corresponding number of data busses implemented. The following table summarizes this relationship based on RP04 or RP06 device transfer rates.

Memory Interleave
and KL10 Bus Mode

Number of Channels

Transferring Simultaneously

1	2	4
2	5	6

This table presumes no "overruns". Should these occur they would be handled by TOPS-10 in the same manner as direct to memory channels. It should be noted that for typical data transfer rates of under 1000 blocks per second there is no measurable conflict between instruction execution and I/O transfer.

Console/Diagnostic Computer

One of the more significant features of the KL10 is the PDP-11-based Console/Diagnostic Computer.

The PDP-11 communicates with the KL10 through a DECsystem-10/PDP-11 interface. The interface allows data transfers between the PDP-11 and the KL10 to take place simultaneously. On the PDP-11 side of the interface, data has direct memory access over the PDP-11 UNIBUS in 8- or 16-bit bytes. On the KL10 side of the interface, data may exist in any size byte up to 36 bits in length.

The PDP-11 serves as the console device for the KL10 processor and includes a TU56 DECTape for loading and diagnosing the PDP-11 and back-up for the RP04 which normally loads the KL10's micro-code memory. An LA36 DEC-writer-11 provides operator/monitor communication.

As a diagnostic computer, the PDP-11 can examine the data paths and the control logic of the KL10 via a separate diagnostic bus, even if the KL10 is completely inoperative. Other features of the diagnostic computer allow all data busses to be checked. Remote diagnostic checking over telephone connection can also be performed via the PDP-11 computer.

The console/diagnostic computer is a state-of-the-art concept to provide the DECsystem-1090 and DECsystem-1099 with greatly improved mean-time-between-failure and reduced mean-time-to-repair. The KLINIK diagnostic system permits remote diagnosis of system failures via the console/diagnostic computer.



System Integrity Features

Power Fail

If system power fails, a power failure detection circuit senses the condition and causes an interrupt. The interrupt can trigger the operation of a program which saves all valuable registers so that the system can be restarted in a minimum amount of time.

On the KI10 and KL10 (through the PDP-11 console computer) an automatic restart capability has been added to resume normal operations in the event of a power outage. All three phases of AC power are monitored. Low voltage on any phase will initiate a sequence of power-down operations. A program-selectable automatic restart capability is provided to allow resumption of operations when power returns. Alternatively, a manual restart may be used.

Temperature sensors strategically placed within the equipment detect high temperature and low air flow conditions and cause power shutdown. This, in turn, initiates the power failure interrupt.

Error Handling

The TOPS-10 Monitor records errors that occur in the slave CPU (on a dual CPU system) or peripheral subsystems and notifies the operator. If the error can be corrected then users affected simply continue running. Alternatively, the operator may request that the faulty device be detached in which case the user job will be notified and continued or restarted as required. If the error detected is in memory then TOPS-10 will attempt to move its own data, if any, out of the failing memory. In the case of disks, the system will attempt to migrate any swapping space to another unit to prevent loss of jobs. TOPS-10 also provides diagnostics to facilitate on-line maintenance of a particular device concurrent with user mode operation.

Multiprocessor Systems

Dual CPU System

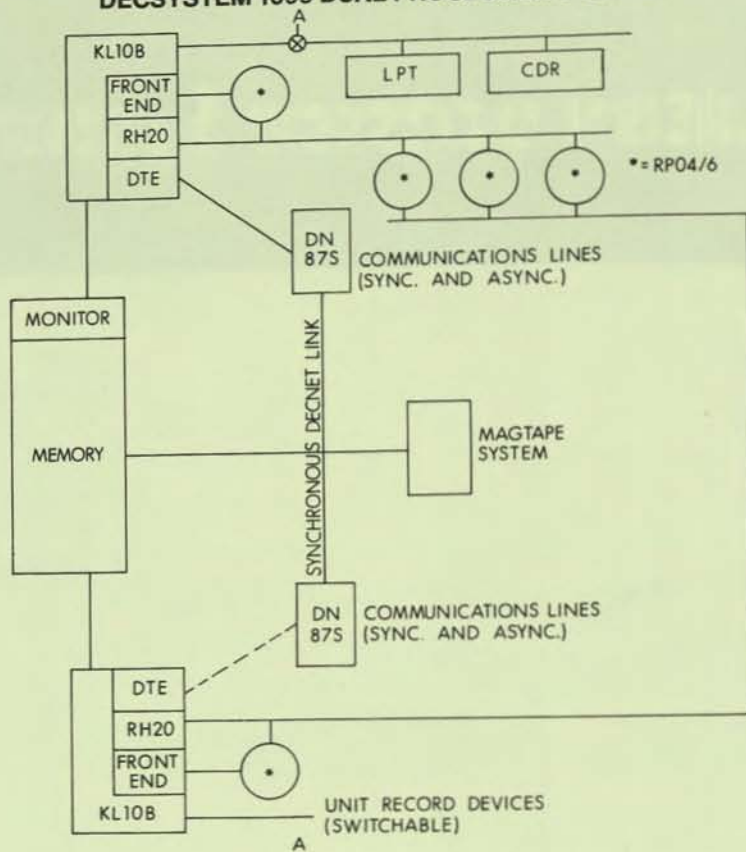
Multiprocessor systems, built around all of DIGITAL's current CPU's, offer attractive, economical solutions needed for capacity and availability.

In these configurations, two processors divide responsibilities as master and slave. The master runs the TOPS-10 operating system and handles all scheduling and I/O. One processor can take over the other's role through the use of strategically-placed bus switching hardware, which allows peripheral devices to be under the control of either processor. If the master malfunctions, the system stops and an operator can have it back into action in under five minutes by reconfiguring all I/O to the slave, which then becomes the master. Users just temporarily have somewhat less computing power to work with, but can still get their jobs done. Minutes after a CPU malfunction that could keep a single system down for hours, the dual processor configuration can be back in action. Redundant disks and communications subsystems can provide increased capacity while all components are fully operational and back-up in case of an isolated controller or subsystem malfunction.

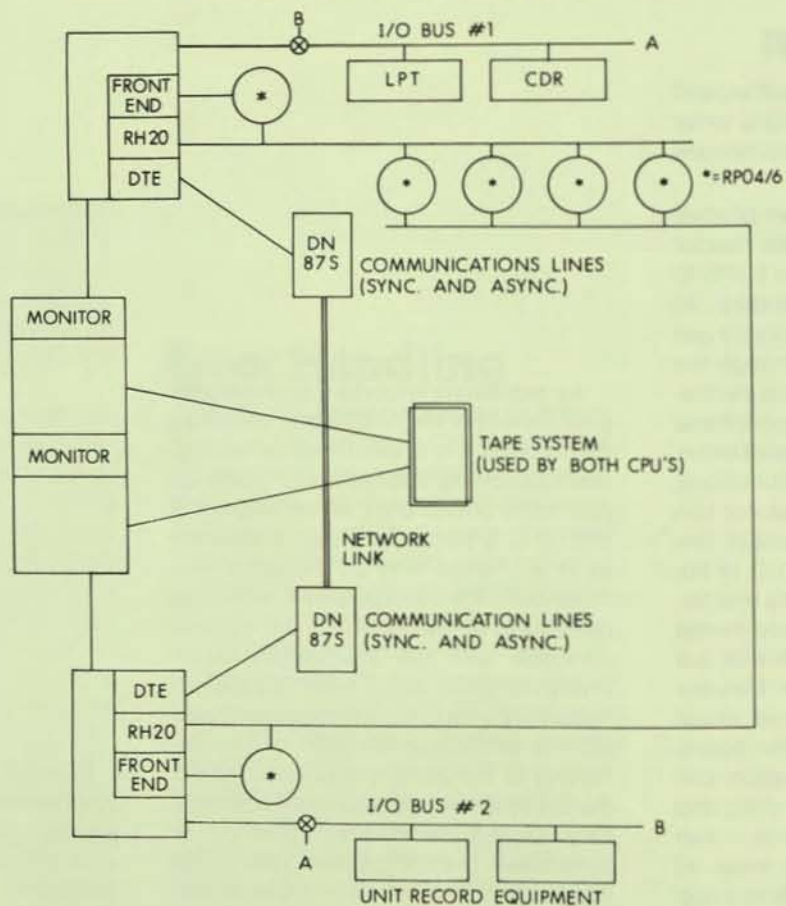
An extremely important and valuable side benefit of the redundant, switched configuration is system maintainability. Malfunctioning subsystems, (such as disk controllers, etc.) which require a CPU to fully diagnose, repair, and certify as operational, may be included in a "maintenance configuration" while the remaining components of the system continue with the production tasks. Users need not suffer from any lack of capability until scheduled maintenance activity or "non-prime time" allows the testing of the complete system. When the subsystem or component is released, fully tested, from Field Service, a scheduled reconfiguration (providing minimum impact on users) can be accomplished, providing full capability again.

Dual CPU/peripheral configurations can be initially installed or, through the addition of appropriate redundant system components and bus switching hardware, can provide attractive upgrade/migration paths for existing systems.

DECSYSTEM 1099 DUAL PROCESSOR SYSTEM



DUAL DECSYSTEM 1090 SYSTEM



Peripherals

Disk Systems	20
DAS33 Buffered Data Channel	22
DEctape Transport	23
Magnetic Tape Systems	24
Hard Copy Equipment	26
Card Readers	26
Line Printers	26
Terminals	28

Disk Systems

There are four disk systems available on the DECsystem-10. The RHS04 system is designed to provide the DECsystem-10 with a fast-access, high transfer rate swapping system which greatly enhances system performance and load handling capability. The DECsystem-10's three disk pack systems offer rapid access, on-line storage, and large capacities at low cost. Modularly expandable, a DECsystem-10 allows up to four controllers each with eight drives giving a total capacity in excess of 6.3 billion ASCII characters. In addition to storage, a disk system may be used as either the primary or secondary swapping system.

The RTP06 is DIGITAL's newest disk system. It operates through a controller which transfers data directly to and from memory via the KL10 internal channels. Since the controller provides overlapped positioner operation, the TOPS-10 operating system will simultaneously position two or more disk drives, shortening the effective access time and increasing throughput.

Each RP04 or RP06 drive configuration can have an optional dual Mass Bus® channel connection which allows access and data transfer via two paths for each individual drive.

The RH04 and RP06 offer a high level of data reliability. A phase locked loop clock system and MFM (modified frequency modulation) recording provide the latest in reliable reading and recording techniques. In addition, error detection and correction hardware provides information as to the position and pattern of any error burst up to 11 bits within the data field. Correction of any data field errors is then accomplished under TOPS-10 software control.



#

DAS33 Buffered Data Channel

The DAS33 is intended to replace DF10 data channels in large DECsystem-10 configurations containing many high-speed peripherals where memory transfer throughout is a critical consideration in a peak memory contention environment.

- Hardware and software compatible with the standard Data Channels.
- 256-word data channel buffer reduces memory contention for high-speed, direct-to-memory peripherals, increasing memory transfer throughput.
- Control word and data word "look-ahead" logic
- 22-bit memory addressing (four million words of 36-bit memory)

The DAS33 has been developed by the Advanced Systems Group to reduce peak memory contention and to increase memory transfer throughput for the new generation of high-speed peripherals such as swapping disk pack systems. It is fully hardware and software compatible with the existing data channels. By incorporating a 256-word buffer (equivalent to two disk sectors), block transfers are averaged over available memory bandwidth. This results in a minimization of memory contention during peak memory transfer and an increase in usable memory transfer throughput for high-speed peripherals. Throughput is further enhanced by the "look-ahead" feature for control and data words.

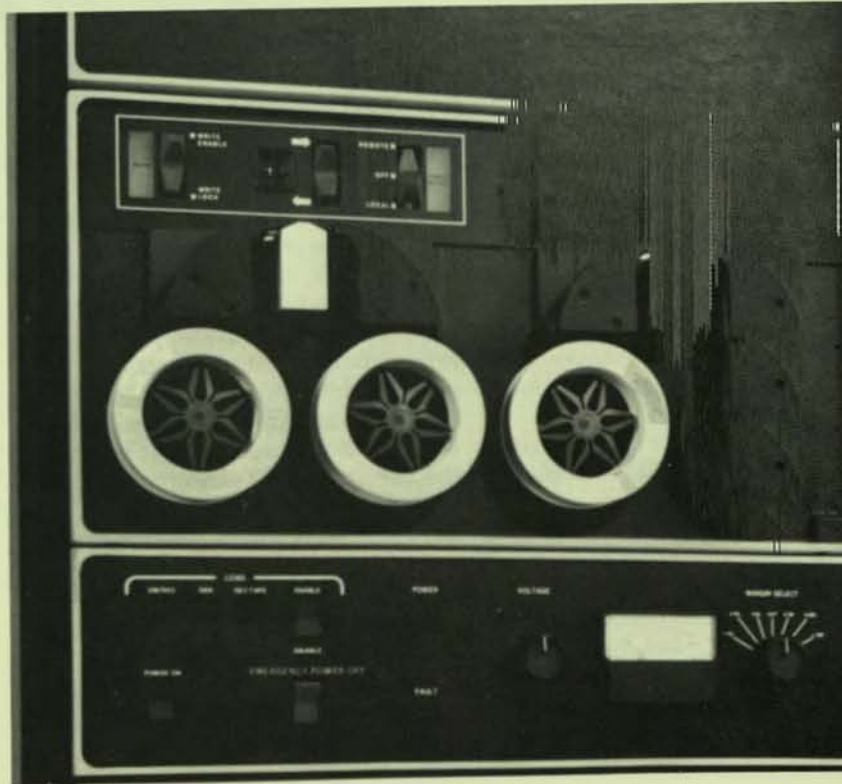
The DAS33 is recommended for large DECsystem-10 configurations having swapping and disk pack subsystems. By minimizing memory contention (with the two-disk sector buffer) and by providing control and data work "look-ahead", the DAS33 will increase the system memory throughput to/ from these high-speed devices. Its 22-bit-wide memory addressing but enables I/O controllers to address up to four million (36-bit) words of memory.

The DAS33 includes its own 19-inch cabinet and memory bus cables. One DAS33 is needed for each controller.

DECtape Transport

Digital Equipment Corporation's popular computer data storage medium, DECTape, is available in a dual-transport version. This fixed-address, bi-directional magnetic tape storage system provides random access for high-speed reading or writing of files on 260 foot reels of $\frac{3}{4}$ -inch-wide magnetic tape contained on a reel less than four inches in diameter. Redundant recording (each bit of data is recorded on two separate tracks) assures high reliability and eliminates the need for character parity checking. In addition, block check summing insures data integrity.

DECTape does not require the rewind and count record operations or the writing of separate header records typical of ordinary magnetic tapes. The high reliability and convenient size make DECTape the ideal medium for program storing and transportation.



Magnetic Tape Systems

DIGITAL provides a complete range of magnetic tape systems from the low-cost THU16 to the mid-range TU70/TU71, to the high-performance TU72.

The THU16 subsystem provides 200, 556, 800 bpi (NRZI) and 1600 (Phase Encoded-PE) capability at low cost. With a tape speed of 45 ips, these transports exhibit a 72K character per second maximum transfer speed. For systems with moderate duty requirements, the THU16 is an excellent price/performance choice.

The TU70 series of high-speed magnetic tape drives represents the latest state-of-the-art design in tape transport technology.

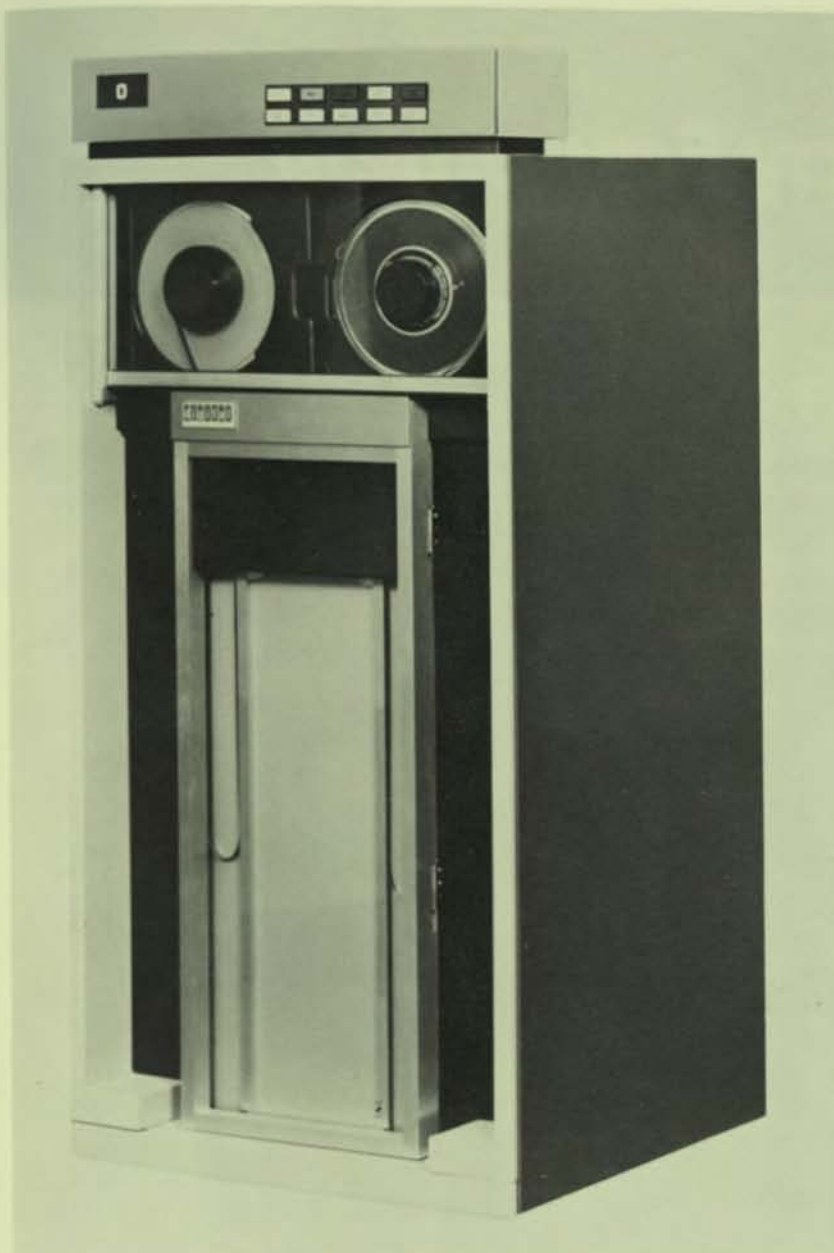
The TU70 provides 800 bpi (NRZI) and 1600 bpi (PE) capability at 200 ips, resulting in a maximum transfer rate of 320K characters per second. The TU71 provides 7-track capability at 220/556/800 bpi (NRZI) at 200 ips for a maximum transfer rate of 160K characters per second.

For very high performance systems DIGITAL offers the TU72, providing faster processing, fewer tapes, faster dumps, and improved error handling for DEC-10 sites. This 6250 bpi drive provides 3 to 1 recording density compression of an equivalent 1600 bpi drive. Using the Group Coded Recording (GCR) technique the higher density provides more capacity per tape, higher data rates and better error correction.

Magnetic Tape Systems

The TU70-series drives incorporate such features as automatic threading and loading of 5-, 8½- and 10½-inch reels as well as industry-standard cartridges. Dynamic amplitude control during read operations improves performance during 1600 bpi operations by reducing read/write errors, avoids the need for preamplifier adjustments, and allows optimum performance with different tape brands on the same drive. All TU70 tape drives are equipped with an automatic reel hub to speed and ease tape loading and alleviate reel slippage. An analog capstan control provides constant motor drive control during read/write and rewind for smoother operation and greater drive-to-drive compatibility, while a velocity feedback reel control provides for protection and velocity sensing of the tape loop in the vacuum columns to minimize reel overspeed; this prevents tape matching, results in longer tape file, and greater machine reliability. A linear high-speed rewind mechanism reduces rewind time and lessens stress on the reel drive system resulting in smoother operation. A tape storage pocket is provided for convenience.

The controller handles up to 8 tape drives and provides direct access to the DECsystem-10 memory via an integrated data channel. The controller includes such features as improved phase-encoded and group coded recording error correction techniques for fewer permanent and temporary read errors. A read-only memory control for incorporation in microprogram diagnostics, monolithic MSI design, for greater reliability, loadable in-line microprogram diagnostics, programmable maintenance memory, adjustment-free read detection, and a radial interface which allows each individual drive to be maintained or physically removed without effecting the operation of other units.



TU70/TU72

Magnetic Tape

	TU10W	TU16	TU70	TU71	TU72
Tape Speed	45 ips	45 ips	200 ips	200 ips	125 ips
Transfer Rate at:					
200 bpi	9 K char/sec	9 K char/sec	—	40 K char/sec	—
556 bpi	25 K char/sec	25 K char/sec	—	111.2 K char/sec	—
800 bpi	36 K char/sec	36 K char/sec	160 K char/sec	160 K char/sec	—
1600 bpi	—	72 K char/sec	320 K char/sec	—	200 K char/sec
6250 bpi	—	—	—	—	780 K char/sec
Recording Technique	NRZI	PE/NRZI	PE/NRZI	NRZI	PE/GCR
Nominal Inter-record Gap:					
9-track	0.6 inches	0.5 inches	0.6 inches	—	0.3 inches
7-track	0.75 inches	—	—	0.75 inches	—
Rewind Time (2400 ft)	195 seconds	3 minutes	45 seconds	45 seconds	55 seconds

Hardcopy Equipment



Card Readers

Designed to meet varying throughput requirements and provide quiet, reliable trouble-free operation, the CR10 card readers accept 80-column EIA-ANSI standard cards. For fast throughput, the user can choose the console model CR10-E which processes 1200 cards per minute or the table model CR10-D which processes 1000 cards per minute. The CR10-F card reader is a lower-cost table model which operates at 300 cards per minute, but utilizes the same excellent features as the high-throughput model.

Line Printers

The LP10-F and LP10-H are drum-type line printers designed with emphasis on output quality and operating reliability. The two-speed capability of the 64-character LP10-F and the 96-character LP10-H increases the printing quality for both low-speed operation and multi-part forms handling.

The LP10-F operates at 1250 or 925 lines per minute and the LP10-H at 925 or 675 lines per minute. Both line printers feature optical vertical format unit using 12-channel tape, and a quick-change drum which allows the character set to be changed in just 5 minutes. Both printers are available with either an EDP or Scientific character set for the quick-change drum.

For user sites requiring high-grade print quality, flexibility of printing fonts, heavy print volumes and high reliability from a horizontal font-motion line printer, DIGITAL offers the LP100. The LP100, an impact printer using a Charaband* available with a 64-character EDP character set font on one side of the band and a 96 character set on the other. The Charaband* can be reversed by the operator in about one minute. The LP100 stores a stream of up to 132 characters in a print line buffer and upon command prints the contents of the buffer and advances the forms as specified.

*Trademark of Dataproducts Corporation



Line Printers

PRINTERS	TYPE	PRINT SET # CHARACTERS	MAXIMUM SPEED LINES/MINUTE	MAXIMUM COLUMN WIDTH	PAPER TYPE	# PARTS OF PAPER FORMS
LP10-F	impact	64	1250/925	132	std.	6
LP10-H	impact	96	925/675	132	std.	6
LP100	impact	64/96	1200/900	132	std.	6

Terminals

Each DECsystem-10 is capable of supporting up to 512 interactive terminals. A wide variety of EIA and 20 mA Current Loop compatible terminals is supported by the DECsystem-10.

The LA36 DECwriter II is designed to be industry's lowest-priced, best performing, most reliable 30-cps impact teleprinter. DECwriter II is packed with capability: it's fast—a true 30-cps machine; it's quiet—you'll almost forget it's there; it has many people-oriented design considerations you wouldn't expect in a cost-conscious product; it's versatile—you can use 6-part forms and even standard 132-column line printer paper; and it's built for a long, reliable lifetime of heavy use.

The LA35 Receive Only terminal is an LA36 without the input key board.

The LA37 APL terminal is really three terminals in one. It provides the user with the choice of either full APL, full upper/lower case ASCII, or upper case TTY in one reliable, easy-to-use hard-copy terminal.

The LA180 DECprinter I is an inexpensive 7x7 dot matrix 132 column printer that prints at 180 characters per second. It features the 128 character ASCII upper-lower case print set, an extensive array of human engineering features, and may be used either locally or as a remote printer hard copy unit.

The DECprinter I extends the field-proven technology of the LA36 DECwriter II into applications demanding higher speed capabilities.





Hardcopy Terminals

CHARACTERISTICS	LA35 RECEIVE ONLY	LA36	LA37APL	LA180 RECEIVE ONLY
Print speed (chars./sec.)	10/15/30	10/15/30	10/15/30	up to 180
Print positions per line	132	132	132	132 (with 133 bit buffer)
Character set	96 ASCII upper/lower case	96 ASCII upper/lower case	Full APL; Full ASCII upper case TTY	128 ASCII upper/lower case
Character formation	7x7 dot matrix	7x7 dot matrix	7x7 dot matrix	7x7 dot matrix
Interfaces	20mA standard; EIA optional	20mA standard; EIA optional	20mA standard; EIA optional	Parallel
# of copies	Original +6	Original +6	Original +6	Original +5
Purpose of Unit	A hardcopy output device (e.g. CRT display attachment)	Interactive hard copy terminal	APL Terminal	A medium-speed hard copy unit
Comments	Basically an LA36 without keyboard			Prices include controller/interface to attach to PDP-8 or PDP-11

DECscopes—DIGITAL's VT50 series of character oriented video terminals—eliminate the need for electromechanical devices. They are all simple for operators to master, quieter than mechanical terminals, and inexpensive. The three models—VT50, VT50H, and VT52—differ only in advanced application features.

The VT61 is the first DECscope that can transmit in either block or character mode. It offers over 80 commands, including character and line insertion and deletion, text justification, reverse video and protected forms, and a number of other editing features not found on the VT50 and VT52 DECscopes.

The VT61 is an upper and lower case ASCII video terminal with a 1920 character display and 19-key function pad. However, the VT61 uses a faster microprocessor with larger memory than do the other DECscope models, so it can offer more stand-alone capabilities.



CRT Display Terminals

CHARACTERISTICS	VT50	VT50H	VT52	VT61
Screen format (lines and chars./line)	12x80 (960 chars.)	12x80 (960 chars.)	80x24 (1920 chars.)	24x80 (1920 chars.)
Character matrix	5x7	5x7	7x7	7x8
Character set	64 ASCII upper case	64 ASCII upper case	96 ASCII upper/ lower case	128 char. upper/lower
Keyboard	Standard typewriter	Standard typewriter plus 19-key pad (10 numeric, enter, period & 3 unlabeled function keys)	Standard typewriter plus 19-key pad (10 numeric, enter, 4 censor, period & 3 unlabeled function keys)	DEC standard +19- key pad + 12 user defined keys
Communications interface	20 mA std., EIA optional	20 mA std., EIA optional	20 mA or EIA standard	20 mA std., EIA optional
Data rates (bps)	75-9600	75-9600	75-9600	75-9600
Editing capabilities				
-Erase to end of line	yes	yes	yes	yes
-Erase to end of screen	yes	yes	yes	yes
-Downward scroll	no	no	yes	yes
-Upward scroll	yes	yes	yes	yes
-Insert/delete character	no	no	no	yes
-Insert/delete line	no	no	no	yes
Cursor control	Cursor movement via ESCAPE sequences	Direct cursor addressing	Direct cursor addressing plus cursor keys	Direct cursor addressing
Optional features	No optional attachments	No optional attachments	Electrostatic copier, LA35RO	LA35 LA180 202T modem interface

DATA COMMUNICATIONS SYSTEMS

DECsystem-10 Data Communications also links networks of computers together from both local and remote sites. Data collection stations, remote control stations, remote concentrators, and remote job entry stations are easily tied into a single data network.

An important features of these capabilities, which translates into a direct benefit to users in ease-of-use, is that communications is handled as an integral subsystem of the DECsystem-10 computer system. Communications protocol is handled by the communications front end system, fully in league with the TOPS-10 Operating System. Users need not concern themselves with a "tacked on" communications handler; TOPS-10 interfaces directly with the user environment in a transparent fashion. The powerful and flexible communications capability of the DECsystem-10 is just one of the features of TOPS-10. Users may design and implement their systems by concentrating on the application; the DECsystem-10 Communications products take care of the communications problems.

The DECsystem-10 Data Communications System unifies software and hardware into a capability designed and implemented for flexibility and ease of use. The system provides a wide spectrum of capability to distribute computing power to a continually-growing user community. In concept, the DECsystem-10 Communications System provides a transparent link between remote users and the central site. Because of this transparency, users, at any location, may work with the same facility as those physically located at the central site, with the same set of commands and machine resources used at the central site.

System Concepts	34
Asynchronous Communications	34
Synchronous Communications	34
Synchronous vs. Asynchronous Transmission	35
Remote Communications	36
Network Concepts	37
Simple Topologies	37
Complex Topologies	37
Task-to-Task Communication	38
IBM Device Emulation and Termination	38
Communications Products	39
DN87/87S Universal Synchronous/Asynchronous Front End Subsystem	39
Complex Topologies	42
DECnet-10	42
DAS80 Series Remote Station	43
DAS92 Remote Station	45
DAS 61 IBM 2780/3780 Support	46
DAS 62 IBM HASP Multi-leaving Support	46

System Concepts

Asynchronous Communications

Asynchronous communications equipment, with associated terminals, serve a broad spectrum of user needs within the DECsystem-10 interactive environment. These needs include interactive program development, operator control of the system, program production and control, data entry, program and data preparation, interactive problem solving, student instruction, and information storage and retrieval, to name a few. Asynchronous communications is used to link the DECsystem-10 with terminal equipment of three types: hard copy, such as the LA37 or LA36 DEC-writers, CRT terminals such as the VT52 DECscope, and buffered terminals, such as the new VT61.

Local terminals within 1500 feet of the computer site can be connected over dedicated hard-wired lines. Remote terminals located beyond this distance are connected over dedicated or dial-up telephone lines.

Speeds of asynchronous hard-copy terminal equipment generally range from 10 to 30 characters per second. CRT terminals range in speed from 10 to 960 characters per second, depending upon their communications interface and type of line used for transmission.

The goal of an interactive terminal system is to match all users to the system in such a way that they feel the terminal is identified closely with the computer. In this way, the machine becomes an easy and natural extension of each application. To do this, the computer must be a willing partner in the interchange so that the user feels that he is in control of the terminal, and not the reverse. Typically, a user types input sporadically, occasionally makes mistakes, and reads fast on output. The Data Communications system for the DECsystem-10 has been engineered to deal with these user characteristics.

The system, which includes hardware and software elements, accepts characters at uneven rates, allows for single character or line correction of mistakes, and allows the user to type at his own speed—even typing ahead of the computer's response to his characters. When users are entering large volumes of data and need no prompting, the system accumulates data and anticipates ongoing entries so that the user can continue to type despite fluctuations in the system's response.

Features of DECsystem-10 asynchronous communications include the ability to delete one or more characters, delete entire lines, retype lines and characters, interact on a character or complete line basis (depending upon the application), suppress unwanted output, and make use of prompting messages to call for input. Terminals may communicate with the system administrator or operator at the central or remote sites as well as with other terminals.

Synchronous Communications

DECsystem-10 synchronous communications provides error-free, high-speed paths between the central computer and remote stations or other computer systems. DECsystem-10 synchronous equipment controls transmission over high-speed synchronous lines. This transmission is on a message basis in contrast with the character-by-character basis of lower-speed, asynchronous transmission.

Full duplex protocol software supplied with the DECsystem-10 makes efficient use of high-speed transmission in both directions simultaneously on a full-duplex line. Front end hardware and software handles hardware control, message formatting, and message acknowledgements. The data transmission is "pipelined", a technique which increases line efficiency by fully overlapping the acknowledge-continue signals.

Transmission errors are detected using block-oriented techniques including longitudinal data checks. Data errors are corrected through retransmission of the erroneous block. Hardware is available to interface to a broad range of communications modems.

Synchronous vs. Asynchronous Transmission

Asynchronous transmission is employed to transmit data on a character-by-character basis; synchronous transmission is preferred for moving large blocks of data. Asynchronous transmission, at 2400 baud, is limited to conveying no more than 240 eight-bit characters per second; synchronous transmission, at 2400 bits/second, can convey 300 eight-bit characters per second. This difference comes about because synchronous transmission uses independent, block-oriented clock synchronization, while asynchronous transmission re-establishes its character-oriented clock synchronization with each character. Thus, in the asynchronous case, each eight-bit character requires, minimally, one start bit and one stop bit.

The most important distinction between asynchronous and synchronous methods of transmitting data is the means of error detection and recovery. Error detection for asynchronous devices is normally accomplished by insertion of a parity bit into the data field of each character transmitted. Half-duplex (local character echo) receivers will typically check this parity bit and notify the sender of the existence of an error. In full-duplex systems, such as the DECsystem-10, the character echo is generally provided by the communications computer (DN87, DAS80, etc.) to which the user is immediately attached. In this case, transmission error detection and corrective action are thus the responsibility of the sender, who simply compares what was sent with what the computer says it received. The communications computer which echoes the character to the user then guarantees that it will deliver that character, without error, through the network, to the appropriate DECsystem-10 host.

This mode of error detection is highly efficient and fully acceptable for interactive, terminal-oriented systems, but is less efficient and less than acceptable for message-oriented communications with other computers. Synchronous communication normally uses a block-oriented error detection technique, such as the Cyclic Redundancy Check (CRC-16) polynomial. This technique is most efficient for block-oriented data transmission and for this reason, communications with remote computers is accomplished using synchronous data transmission.

Remote Communications

Prior to the current line of DECsystem-10 remote communications products, each remote user had been individually linked to the computer center by separate long-distance telephone lines.

The only device that the remote user had available at his location was the terminal; he was able to utilize available devices at the central station, but he could not obtain output other than his terminal output at the remote site. Either the user had to travel to the central station to obtain a listing, or he had to have the listings delivered to him. However, with remote communications hardware and software, listing files and data can be sent via a single synchronous full-duplex line to a small remote computer. That remote computer in turn services many remote peripherals, including terminals, card readers and line printers. This eliminates the need for the user to travel to the central site to obtain his output. The remote computer and its associated peripherals constitute a remote station.

Remote station use of the central computer is essentially the same as local use. All sharable programs and peripherals available to local users at the central computer station are also available to remote users. The remote user specifies the resources he wants to use and, if available, these resources are then allocated in the same manner as to a local user. In addition to utilizing the peripherals at the central station, the remote user can access devices located at his station or at another remote station. Local users at the central station can also make use of the peripherals at remote stations. Therefore, by specifying a station number in appropriate commands to the operating system, each user of the DECsystem-10 is given considerable flexibility in allocating system facilities and in directing input and output to the station or DECsystem-10 of his choice.

The DECsystem-10 allows for simultaneous operation of multiple remote stations. Software provisions are incorporated in the operating system to differentiate one remote station from another. By utilizing peripheral devices at various stations, the user is provided with increased capabilities. For example, data can be collected from various remote stations, compiled and processed at the central station, and then the results of the processing can be sent to all contributors of the data.



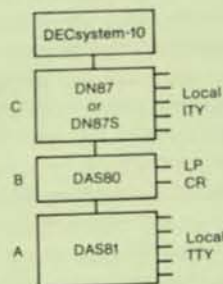
Network Concepts

Complex Topologies

This is the name given to a group of general network building capabilities which, when put together, allow the user to make many different cost, performance and high availability trade-offs so that his network may be configured in a way which is very much optimized for him and therefore very favorable to his environment. These building block capabilities are:

Route-thru

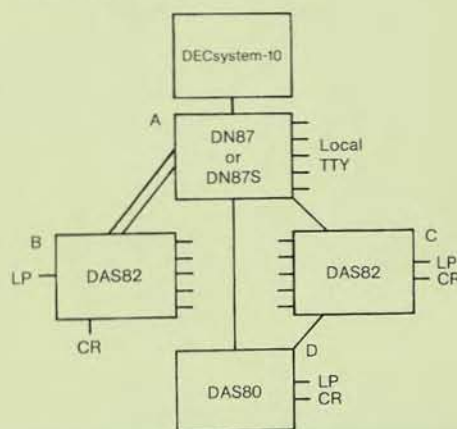
This is the ability for a Remote Station to send information to a DECsystem-10, via the Universal Front End, to which it is only indirectly attached—i.e., indirectly through another Remote Station or Universal Front End. This is most easily illustrated in the following picture:



Information flows from A to C only by being routed through B. This is a very useful price performance trade-off since the cost of a communications line from A to B and from B to C may be substantially cheaper than the cost of lines from A to C and from B to C.

Multi-pathing

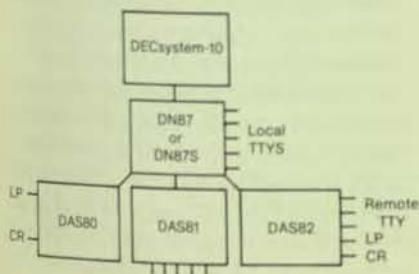
This is the ability to have more than one path between nodes in the network. In general, these links are automatically load leveled depending upon the relative speed and error rate of the lines. This is best illustrated by the following diagram:



The two links between A and B will be automatically load leveled with data flowing through each link. This may be a desirable configuration for high availability systems with, for example, one link via microwave and one by ground link. This would allow degraded operation if either link went out for any reason. There are also two routes from D to A. The direct link could be a high speed link, while the link routed through C could be lower speed and used as hot backup to the direct link. Again, both of these configurations have desirable benefits that the user, in building his network, can pick and choose from depending upon his specific needs and desires. The ability exists to pick and choose rather than having a very rigid set of network configuration rules.

Simple Topologies

The most basic network topology is a point-to-point, or star topology. It is best illustrated by the following diagram.



Multiple Host Support

If there is more than one DECsystem-10 in a network, the Universal Front Ends on the DECsystem-10s may be hooked up to each other via a synchronous link, or the Remote Stations may be hooked up to each Universal Front End. This allows Teletype users anywhere in the network to select on an individual basis, before they LOGIN, which DECsystem-10 they wish to LOGIN to. This is accomplished by means of the SET HOST command.

Dynamic Topology

This feature allows individual nodes in a network to go down or come up without affecting the remainder of the network. The exact implications of an individual node going down depend on the type of node and its network functions, but any new node may be added to the network dynamically.

Task-to-Task Communication

DECnet-10 provides the capability to have tasks or jobs in different and geographically removed processors communicate with each other in a straightforward manner. These jobs appear to each other as mere I/O devices much like a magtape. Jobs in different DECsystem-10s may communicate with each other, or DECsystem-10 jobs may communicate with other DECnet systems, such as RT-11, RSTS, RSX-11 M, D, & S, IAS, and RTS-8 when they are equipped with DECnet software.

This interprocessor communication allows only for jobs to talk to each other, not for terminal concentration. It does not directly allow for remote file access, remote device access or program sharing. It does, though, present a mechanism or a building block which could be used to provide those features.

IBM Device Emulation and Termination

Through the use of IBM device emulation the DECsystem-10 can be attached to an IBM 360/370 computer via a synchronous link. The DECsystem-10 will be treated by the IBM system as if it were an IBM 2780 or 3780 or HASP multi-leaving remote station. This facilitates job submissions and file transfers back and forth between the two computer systems.

Through IBM device termination IBM 2780's, and HASP multi-leaving work stations can be used as Remote Job Entry stations to enter batch jobs into the DECsystem-10; batch jobs may be placed in the card reader of the 2780, 3780 or HASP work station, read into the DECsystem-10, executed and have their output returned to the 2780, 3780 or HASP work station line printer.

Communications Products

DN87/DN87S

Universal

Synchronous/

Asynchronous

Front End Subsystem

The DECsystem-10 data communications products are based on the DN87 Universal Synchronous/Asynchronous Communications Front End Subsystem. The DN87 is configurable in asynchronous only mode (up to 112 asynchronous lines), synchronous only mode (up to 12 synchronous lines) or with a mixture of synchronous and asynchronous lines in the same DN87. These three modes allow the DN87 to be configured very cost effectively in a wide variety of customer specific configurations. This great flexibility is the essence of the DN87 Communications Subsystem.

The DN87 is capable of synchronous communication with the DAS80 series Remote Stations, the DAS92 Remote Station and the DC72NP Remote Station. These allow Remote Job Entry, Remote Concentration of interactive Teletype lines or a combination of the two. The DN87 with these Remote Stations supports complex topologies such as multi-pathing, route-thru, and multiple host support. With DECnet-10, it is possible to have a DECsystem-10 communicate with any DECnet system—for example RSX-11M, RSX-11D, RSX-11S, IAS, RT-11, RSTS/E, RTS-8, DECsystem-10 or DECSYSTEM-20 if equipped with DECnet software.

The DN87S has the same functionality as a DN87 except that it is attached to the DECsystem-10 via the DTE10 interface rather than the DL10. Up to three DN87S's may be attached to the DTE10 interface. The DN87 and the DN87S require TOPS-10 version 6.03 or later monitor release.

Asynchronous Functionality

On an eight-line group basis, the DN87/87S is capable of terminating 20 mA current loop, local EIA or EIA with full modem control type of asynchronous line/terminal interfaces. On an individual line basis the DN87/87S asynchronous lines can be:

- ASCII Teletype-compatible code or 2741 EBCD or Correspondence Code.
- Full duplex with echoplex (i.e. echo generated by computer) or full-duplex with local copy (simulated half-duplex).
- Program selectable line speeds of from 50 through 9600 baud.
- Split transmit/receive speeds.
- Automatically baud rate detected for 110, 134.5, 150 and 300 baud lines.

Some off-loading of the DECsystem-10 host is accomplished, in that the DN87/87S does the majority of the echoing for asynchronous lines. It does not echo special characters, nor does it echo when the user is in character-at-a-time mode (e.g., DDT—when an individual character can be a command). The DN87/87S also does fill character generation.

Synchronous Functionality

The DN87/87S is capable of terminating EIA and/or current-loop type synchronous links. The line speeds may be 2400, 4800, 7200, 9600, 19.2K, 38.4K or 40.8K baud on an individual line basis. These links operate only in full-duplex with simultaneous bidirectional transmission. The synchronous links use DDCMP protocol for error checking and correction and for point-to-point link control.

These synchronous links communicate only with the DC72NP Remote Station (DC72NP is a software only upgrade of the DC71 or DC72), the DAS80 Series Remote Stations, the DAS92 Remote Station or another DN87/87S. On an individual synchronous line basis the DN87/87S can also communicate on a task to task basis with systems running DECnet software.

Options List

The DN87 is the Universal Synchronous/Asynchronous Front End Communications Subsystem. Software to provide complex topology support is included with the DN87. The DN87 requires the addition of DN81-xx synchronous and/or asynchronous line options.

DN87-DA, DB—DN87 including DL10-C port only. Requires DL10-A with available port. Note that this is what would be ordered with a 1080 or 1060 System Package, which includes a DL10.

DN87-AA, AB—DN87 including DL10-A Communications Interface and one DL10-C port.

DN87S-AA, AB—DN87 with DTE10 interface.

ASYNCHRONOUS LINE OPTIONS

DN81-EA, EB—Asynchronous Expansion Cabinet including one DN81-EC 16-Line Asynchronous Expansion Group. Requires two DN81-Fx 8-Line Terminator Groups to activate the lines.

DN81-EC, ED—Asynchronous 16-Line Expansion Group. This requires two DN81-Fx 8-Line Terminators to activate the lines.

DN81-FA—8-Line Terminators each with 20 mA Current Loop Local Interfaces.

DN81-FB—8-Line Terminators each with EIA Local Interfaces.

DN81-FC—8-Line Terminators each with EIA Full Modem Control Interfaces.

DN81-FD—8-Line Terminators with Integral Auto Answer Modems. (These modems require customer supplied DAA's).

SYNCHRONOUS LINE OPTIONS

DN81-EE, EF—Synchronous Expansion Cabinet. Includes one DN81-H.

DN81-H—Synchronous Line Controller Expansion Line. For data transmission speeds of up to 10K baud and for attachment to EIA RS232C compatible modems.

DN81-J—Synchronous Line Controller. For data transmission speeds of up to 40.8K baud and for attachment to 303-type current mode modems.

NOTE: Option designations are such that if there are two designations the first is 115V/60Hz and the second is 230V/50Hz. If there is only a one option designation that option is NOT power dependent.

Hardware Configuration Guidelines

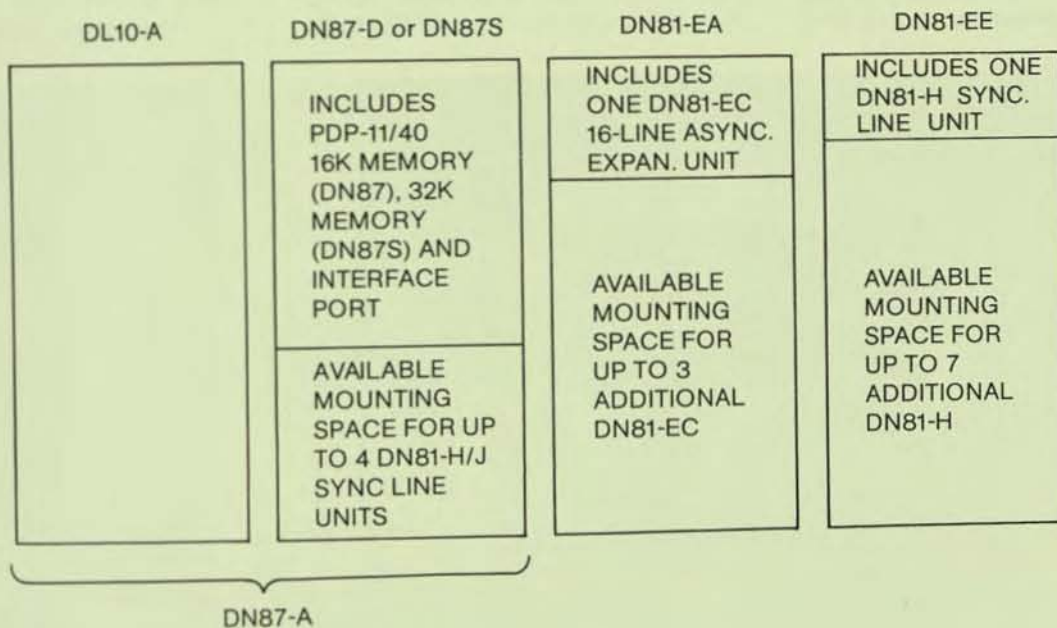
A pictorial representation of the DN87 cabinet arrangements should help to simplify the explanation of the configuration rules.

Asynchronous Configuration Rules

- The DN87 cabinet (DN87-A or DN87-D) has no available mounting space for asynchronous lines.
- If any asynchronous lines are required a DN81-EA Asynchronous Expansion Cabinet must be ordered. This cabinet includes one DN81-EC Asynchronous 16-Line Expansion Group and has available mounting space for up to three additional DN81-EC units. Therefore, the DN81-EA is capable of containing up to 64 asynchronous lines.
- A maximum of two DN81-EA Asynchronous Expansion Cabinets are permitted per DN87.
- Each DN81-EC Asynchronous 16-Line unit requires two DN81-Fx (FA, FB, FC or FD) 8-Line Terminators to activate the lines.

Synchronous Configurations Rules

- The DN87 cabinet (DN87-A or DN87-D) has available mounting space for up to four DN81-H/J synchronous line units.
- If more than four Synchronous Line Units are required a DN81-EE Synchronous Expansion Cabinet must be ordered. This cabinet includes one DN81-H Synchronous Line Unit, and available mounting space for up to seven additional DN81-H units.



LINE MAXIMUMS PER DN87/DN87S

Max. No of Sync. Lines	Max. No. of Async. Lines
0	112
4	64
8	32
12	0

- Proper number of asynchronous lines are made up using the Asynchronous Configuration Rules from above.
- Proper number of synchronous lines are made up using the Synchronous Configuration Rules from above.

Complex Topologies

Complex topologies are available to all DECsystem-10s using the DAS80 Series Remote Stations and the DN87 Front End (includes DC75NP and DAS85) with the 6.03 version of TOPS-10. This means that route-through, multipathing, multiple host support and dynamic topologies (SET HOST monitor command) are all a standard part of the 6.03 monitor and available to those who have the appropriate communications hardware and software configurations.

DECnet-10

DECnet is the set of software products which extend various DIGITAL operating systems so they may be interconnected with each other to form computer networks. The DECnet user can configure a variety of networks, satisfying a variety of constraints, by choosing the appropriate CPU's, line interfaces (and speeds) and operating systems software.

Digital Network Architecture, implemented across a wide range of operating systems and hardware architectures, enables users to build a variety of networks.

DECnet-10 provides interprocessor task-to-task communications for the DECsystem-10. Thus users may communicate on a program to program basis with any other DECnet system through the DN87 front end. This will include RSX-11S, RSX-11M, RSX-11D, IAS, RT-11, RSTS, RTS-8, DECSYSTEM-20 and other DECsystem-10s.

DAS 80 Series Remote Station

The DAS80 Series Remote Stations are a family of remote stations with a wide range of functionality. The DAS80 is a Remote Job Entry Station with a 300 cpm Card Reader, a 300 lpm Line Printer and an LA36 Console. The DAS81 is a Remote Concentrator capable of concentrating up to 32 asynchronous lines. The DAS82 is a combination of the DAS80 and DAS81, facilitating both RJE and Remote Concentration.

When linked to a DN87/DN87S Universal Front End the DAS80 Series Remote Stations will support complex topologies such as route-thru, multipathing, and multiple host support.

Asynchronous Functionality

The asynchronous terminal lines attached to the DAS80 Series Remotes have the same functionality as the asynchronous lines attached directly to the DN87/DN87S.

Synchronous Functionality

The synchronous line is used as a full duplex dedicated line to any one of a DC75NP, DAS85 or DN87/DN87S DECsystem-10 Front Ends. The DAS80 Series Remote Stations are all downline loadable from the DECsystem-10 through the synchronous link.

Unit Record Devices

The Line Printer (LP) and the Card Reader (CR) on a DAS80 or DAS82 Remote Station are available from the DECsystem-10 host in a transparent manner. This means that any user at any terminal can use this remote LP or CR very similarly to how he would use a LP or CR attached directly to the host.

This is accomplished by use of the DECsystem-10 Monitor Command LOCATE which sets the default LP and CR for a specific user.

The LA36 DECwriter, which is included with the DAS80 Series Remotes, can be used as the Operator Controlling Console for the spoolers which are running for the LP and CR at that station. This means that any operator-type requests (e.g., LP out of paper) appear at the remote location with the LP and CR rather than at the host.

Host Interfaces

The DAS80 Series Remote Stations can only communicate with the DECsystem-10 via the DC75NP, the DAS85 or the DN87/DN87S Front Ends. The DN87/DN87S requires the 6.03 or later version of the TOPS-10 monitor, while the DC75NP and the DAS85 require the 6.02A or later monitor. The DC75 Front End will not run under 6.02 or later monitors unless it is equipped with the "NP" upgrade (additional 4K memory, KG11 CRC unit and software).



OPTIONS LIST

DAS80-AA, AB—Remote Batch Station Includes 1 DN81-H, LA36 Console, DAS80-CA/CB, DAS80-LA/LB, PDP-11 and software.

DAS81-AA, AB—Remote Concentrator Includes PDP-11, 1 DN81-H, DN81-EA and software. One more DN81-EC can be added for a total of 32 asynchronous lines.

DAS82-AA, AB—Remote Batch Station and Concentrator. Includes PDP-11, LA36 Console, DAS80-CA/CB, DAS80-LA/LB, DN81-H, DN81-EA and software. DN81-Fx asynchronous interfaces are also required and must be ordered separately.

UNIT RECORD OPTIONS

DAS80-CA, CB—Card Reader, 300 cards per minute.

DAS80-LA, LB—Printer, 300 lines per minute, 132 printing positions, 64 printing characters, EDP character set.

DAS80-LC, LD—Printer, 230 lines per minute, 132 printing positions, 96 printing characters, EDP character set.

ASYNCHRONOUS LINE OPTIONS

DN81-EA, EB—Asynchronous Expansion Cabinet including one DN81-EC 16 Line Asynchronous Expansion Group. Requires two DN81-Fx 8-Line Terminator Groups to activate the lines.

DN81-EC, ED—Asynchronous 16-Line Expansion Group. This requires two DN81-Fx 8-Line Terminators to activate the lines.

DN81-FA—8-Line Terminator each with 20mA Current Loop Local Interfaces.

DN81-FB—8-Line Terminators each with EIA Local Interfaces.

DN81-FC—8-Line Terminators each with EIA Full Modem Control Interfaces.

DN81-FD—8-Line Terminators with Integral Auto Answer Modems. (These modems require customer supplied DAA's).

SYNCHRONOUS LINE OPTIONS

DN81-H—Synchronous Line Controller Expansion Line. For data transmission speeds of up to 10K baud and for attachment to EIA RS232C compatible modems.

DN81-J—Synchronous Line Controller. For data transmission speeds of up to 40.8K baud and for attachment to 303-type current mode modems.

Note: These same line options (DN81-xx) are used on the DAS80 series remote stations, as described here, on the DN87 and for DC76 add-ons.

DAS 92

Remote Station

The DAS92 Remote Station is a low cost remote job entry concentrator that is attached to the DECsystem-10 via one synchronous link to a DN87/DN87S front end. The DAS92 may be configured in one of the following manners:

1. up to 16 asynchronous terminal lines
2. up to 12 asynchronous terminal lines plus a card reader or line printer
3. up to 8 asynchronous terminal lines plus a card reader and line printer

Only one synchronous interface is permitted on the DAS92, therefore the DAS92 must be a sequential node. This means it may not do multi-pathing or route-thru. Data to the DAS92 may be routed from some other node, but no data may be routed through the DAS92 since it is only capable of having one synchronous link.

Options

1. DAS92-AA, AB Basic Unit, including 16K, PDP-8/A processor, VT52 console terminal, desk, ROM for down-line loading, one (1) synchronous line interface and software. Requires at least one (1), but a maximum of five (5) optional units listed below:

(a) DAS92-EA Asynchronous 4-line multiplexer, including line drivers. The DAS92-EA Asynchronous 4-line Multiplexer will accommodate either 20 ma. or EIA lines in any mixture. One out of every four can be full modem control. The baud rate is switch selectable anywhere from 50 to 4800 baud. Maximum of four (4) DAS92-EA's per DAS92-A.

(b) DAS92-CA, CB Card Reader. The DAS92-CA, CB Card Reader is a Digital CR8-F for 80-column (only) cards, and operates at 285 CPM. Maximum of one (1) Card Reader per DAS92-A.

(c) DAS92-PA LA180 Printer. The LA180 operates at 180 characters per second and uses a 96-character set (upper and lower case). The carriage is 132 columns wide. Maximum of one (1) LA180 per DAS92-A.

(d) DAS92-VA, VD LP05 Line Printer, (64-characters). This is the LP05 (LE8-V) and operates at 300 lines per minute using a 64-character set. The carriage is 132 columns wide. Maximum of one (1) LP05 per DAS92-A.

(e) DAS92-WA, WD LP05 Line Printer, (96 characters). This is the same as the DAS92-VA except that it uses the 96-character set (upper and lower case). Maximum of one (1) LP05 per DAS92-A.



DAS61

IBM 2780/3780

FRONT END

The DAS61 Front End allows IBM 2780 and 3780 emulation and termination. This means that IBM 2780's and/or 3780's can be used as RJE stations into the DECsystem-10 or that the DECsystem-10 can emulate (or simulate) an IBM 2780 or 3780 to an IBM 360/370 computer. The DAS61 can handle a maximum of 12 synchronous lines each operating independently with any combination of 2780 or 3780 emulation or termination. The maximum aggregate throughput of a DAS61 is 100,000 bits per second (100K baud). The DAS61-D is interfaced to the DECsystem-10 via the DL10 and the DAS61-S is interfaced via the DTE10. Multiple DAS61's are supported per DECsystem-10.

OPTIONS

DAS61-DA, DB—IBM 2780 and/or 3780 Emulation and Termination Front End. Including DL10-C port and requiring DL10-A with available port. Requires addition of DAS61-H and/or DAS61-J synchronous line units.

DAS61-SA, SB—IBM 2780 and/or 3780 Emulation and Termination Front End including DTE10 interface. Requires addition of DAS61-H and/or DAS61-J synchronous line units.

DAS61-EA, EB—Synchronous Expansion Cabinet for expansion past 4 synchronous lines. Includes no synchronous lines.

DAS61-H—Synchronous Single Line Controller for data transmission speeds of up to 10K baud and for attachment to EIA RS232C compatible modems.

DAS61-J—Synchronous Single Line Controller for data transmission speeds of up to 50K baud and for attachment to 303-type current modems.

DAS61-QC—DAS61 Software Only.

DAS61-QX—DAS78 to DAS61 Software Only Upgrade (prerequisite GALAXY).

DAS62

IBM HASP

MULTI-LEAVING

FRONT END

The DAS62 Front End combines all of the functionality of the DAS61 with support of IBM HASP multi-leaving work stations as DECsystem-10 RJE stations and also allows for the DECsystem-10 to simulate a HASP multi-leaving work station to an IBM host computer. The DAS62 can handle a maximum of 12 synchronous lines each operating independently with any combination of 2780 or 3780 or HASP multi-leaving emulation or termination. The maximum aggregate throughput of the DAS62 is 100,000 bits per second (100K baud). The DAS62-D is interfaced to the DECsystem-10 via the DL10 and the DAS62-S is interfaced via the DTE10. Multiple DAS62's are supported per DECsystem-10.

OPTIONS

DAS62-DA, DB—IBM 2780 and/or 3780 and/or HASP multi-leaving work station Front End. Including DL10-C port and requires DL10-A with available port. Requires addition of DAS62-H and/or DAS62-J synchronous line units.

DAS62-SA, SB—IBM 2780 and/or 3780 and/or HASP multi-leaving work station Front End including DTE10 interface. Requires addition of DAS62-H and/or DAS62-J synchronous line units.

DAS62-EA, EB—Synchronous Expansion Cabinet for expansion past 4 synchronous lines. Includes no synchronous lines.

DAS62-H—Synchronous Single Line Controller for data transmission speeds of up to 10K baud and for attachment to EIA RS232C compatible modems.

DAS62-J—Synchronous Single Line Controller for data transmission speeds of up to 50K baud and for attachment to 303-type current modems.

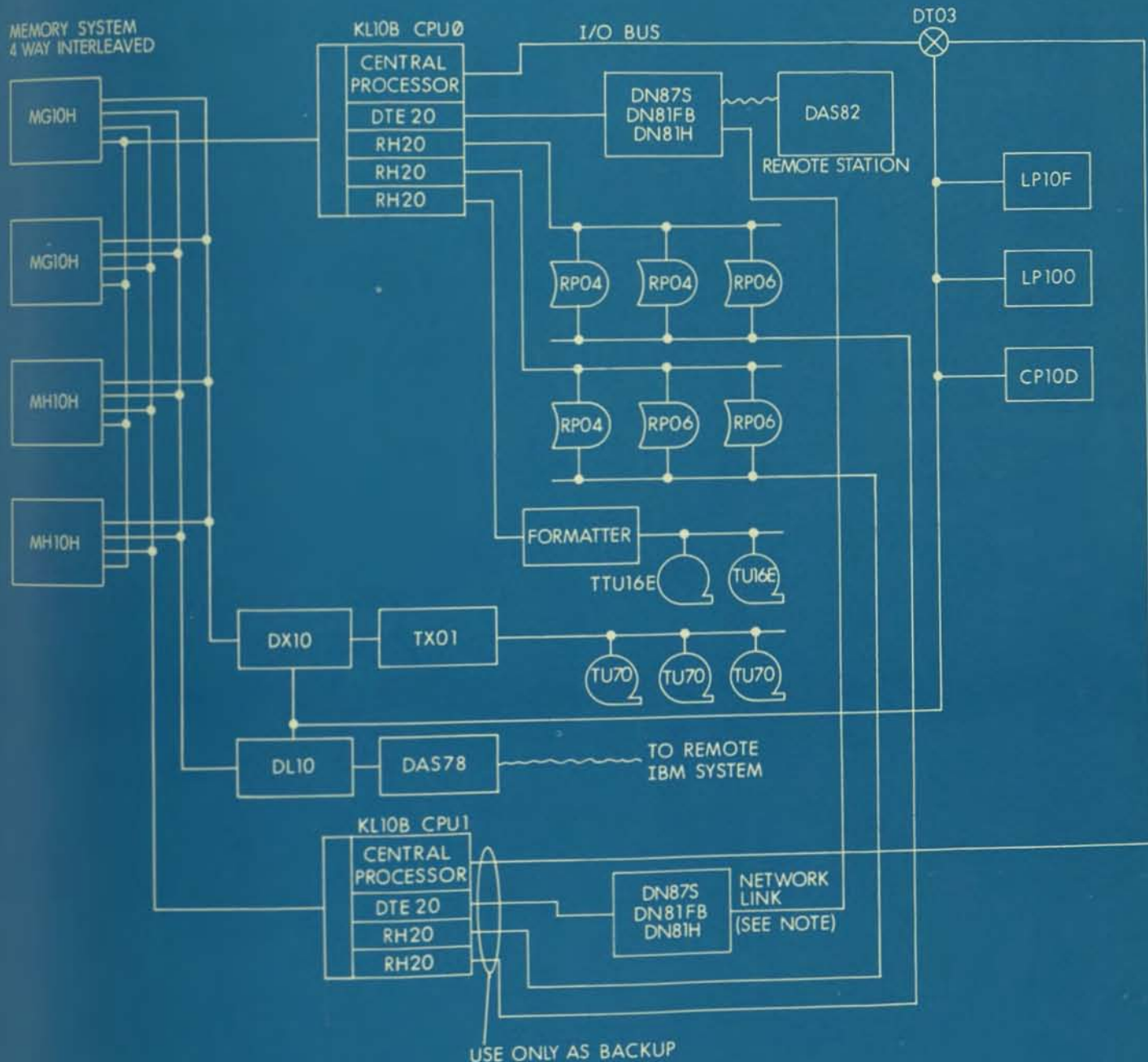
DAS62-QC—DAS62 Software Only

DAS62-QX—DAS78 to DAS62 Software Only Upgrade (prerequisite GALAXY)

DAS62-QY—DAS61 to DAS62 Software Only Upgrade

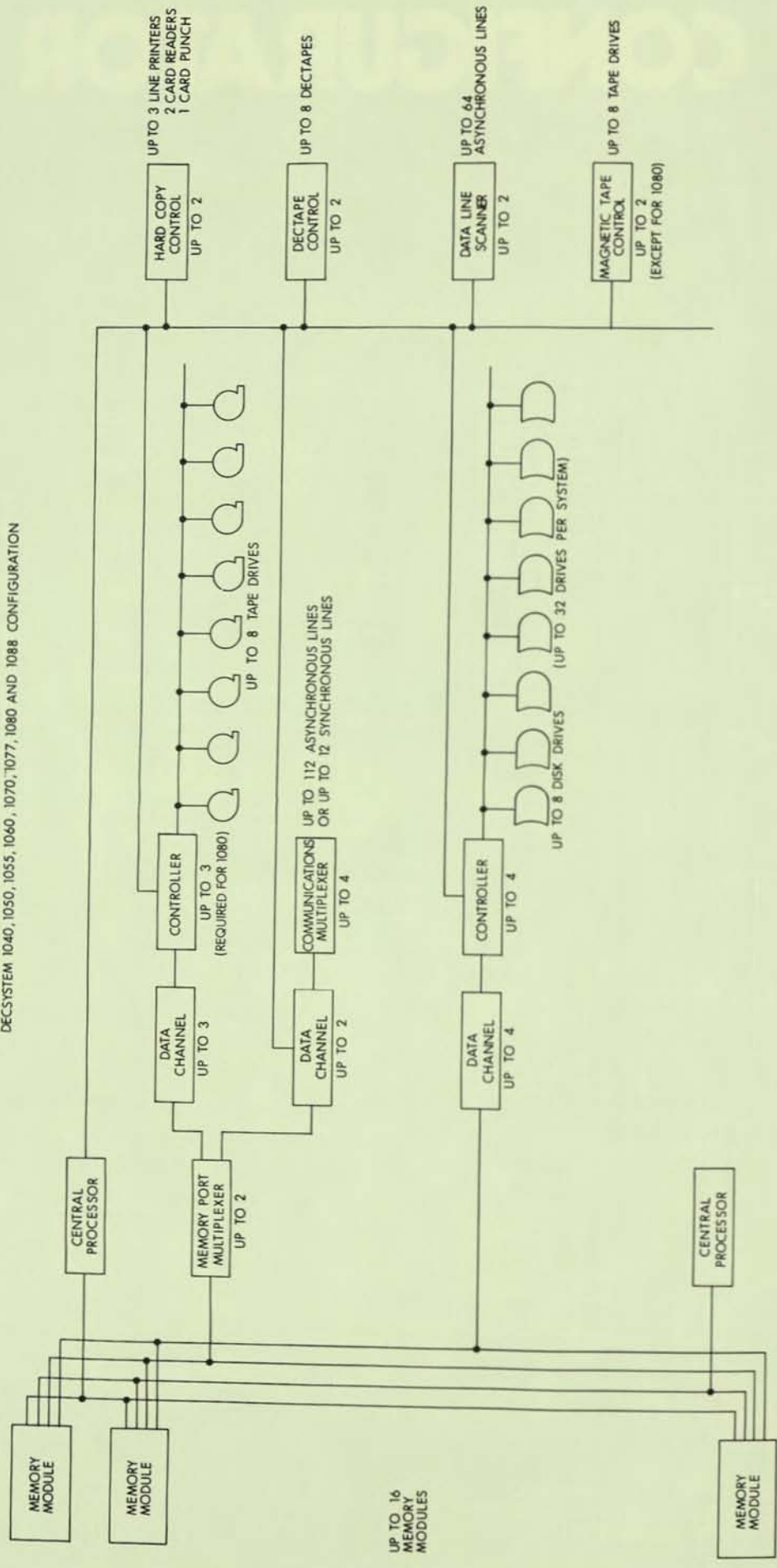
CONFIGURATOR

SAMPLE COMPLEX CONFIGURATION
1090 WITH HIGH AVAILABILITY SWITCHOVER

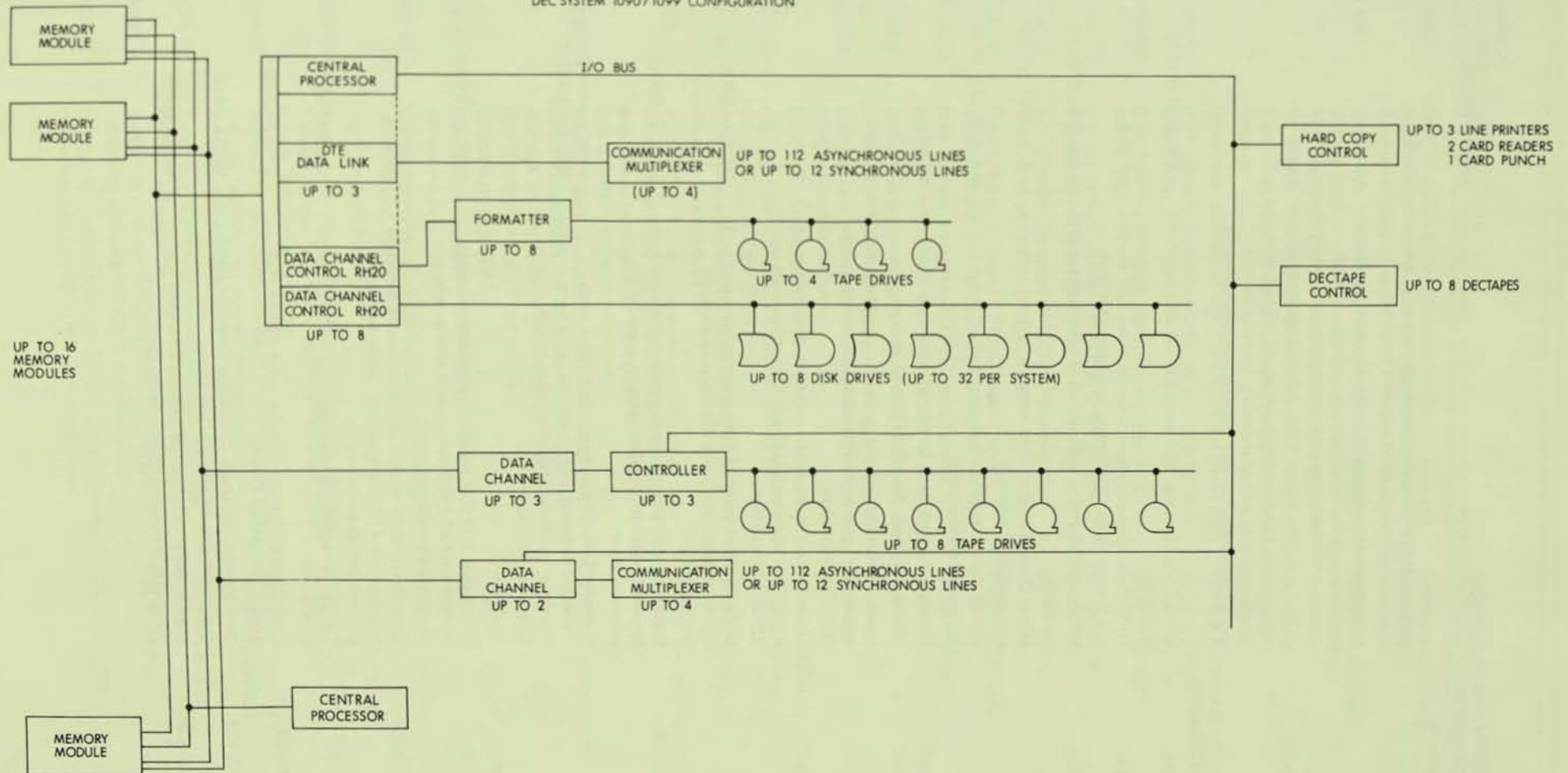


NOTE:
This system would normally run as a 1099 with CPU 0 the master and CPU 1 the slave. The DN87S Subsystem connected to CPU 1 would function as a DAS 81 remote concentrator to the DN87S on CPU 0. Should the master CPU 0 fail, the system would be reconfigured as a 1090 and TOPS-10 reloaded to run on CPU 1. Route-through would be used with the DN87S on CPU 0 now operating as a DAS 81 to the DN87S or CPU 1.

DEC SYSTEM 1040, 1050, 1055, 1060, 1070, 1077, 1080 AND 1088 CONFIGURATION



DEC SYSTEM 1090/1099 CONFIGURATION



OPTION DESIGNATOR 115V/60Hz 230V/50Hz	DESCRIPTION	PREREQUISITE	MAX. NO. SUP- PORTED (NOTE 1)
Central Processors			
KA10-A, KA10-C	KA10 Central Processor; 366 hardware instructions, 10-character-per-second console terminal.	None	2
KI10, KI10	KI10 Central Processor; 378 hardware instructions, memory paging, instruction look-ahead, virtual memory, 10-character-per-second console terminal.	None	2
KL10-BA, KL10-BB	KL10 Central Processor; 395 microprogrammed instructions, business instruction set, high-speed cache memory, virtual memory, instruction look-ahead, 30-character-per-second console terminal.	None	2
Memories and Channels			
MF10-A, MF10-A	MF10 Core Memory; 32K words of 1.0-microsecond memory.	KA10, KI10 or KL10	Note 2
MF10-E, MF10-E	MF10 Core Memory expansion module; 32K words of 1.0-microsecond memory to expand MF10-A.	MF10-A	Note 2
MF10-G, MF10-G	MF10 Core Memory; 64K words of 1.0-microsecond memory.	KA10, KI10, or KL10	Note 2
MG10-HA	MG10 Core Memory; 128K words of 1.0 microsecond memory	KA-10, KI10, or KL10	Note 2
MH10-H	MH-10 Core Memory 256K words of 1.2 microsecond memory	KA-10, KI10 or KL10	Note 2
MC10-D, MC10-D	Memory Access Port; for MD10 Memories	MD10	64
MC10-F, MC10-F	Memory Access Port; for MF10 Memories.	MF10	64
MX10, MX10	Memory Port Multiplexer; provides direct memory access for up to eight FD10 Data Channels using 18-bit address logic.	MC10	3
MX10-C, MX10-C	Memory Port Multiplexer; provides direct memory access for up to eight DF10-C Data Channels using 22-bit address logic.	MC10	3
DF10, DF10	Data Channel; permits data transfer between high-speed devices and core memory; uses 18-bit address logic	MC10 or MX10	8
DF10-CA, DF10-CB	Data Channel; permits data transfer between high-speed devices and core memory; uses 22-bit address logic.	MC10 or MX10-C	8
DAS33	Buffered Data Channel; permits block data transfers between I/O controllers and core memory; contains 256-word data buffer; uses 22-bit address logic.	Avail. memory port MX10 or MX10-C	8
Disk Systems			
RP02-A, RP02-B	Disk Drive; 5.12-million words capacity; average access time 47.5 milliseconds; transfer rate 15 microseconds per word.	RP10 or RP10-C	32
RP03-AS, RP03-BS	Disk Drive; 10.24-million words capacity; average access time 47.5 milliseconds; transfer rate 15 microseconds per word.	RP10-C	32
RP02-P, RP02-P	Disk Pack; additional disk pack for RP02 or RP03 disk drive.	RP02 or RP03	n/a
RP10-CA, RP10-CB	Disk Controller; controller for up to eight RP02 and/or RP03 Disk Drives.	DF10 or DF10-C	3
RP02-CA, RP02-CB	Disk System; includes DF10 Data Channel, RP10-C Controller and one RP02 Disk Drive.	MC10, MX10	3
RP03-CA, RP03-CB	Disk System; includes DF10 Data Channel, RP10-C Controller and one RP03 Disk Drive.	MC10, MX10	3

OPTION DESIGNATOR 115V/60Hz 230V/50Hz	DESCRIPTION	PREREQUISITE	MAX. NO. SUP- PORTED (NOTE 1)
RP04-AA, RP04-AB	Disk Drive; 20.48-million words capacity; average access time 27 milliseconds; transfer rate 5.6 microseconds per word.	RHP04-A, RHP04-B, or RHP04-D	32
RP04-BA, RP04-BB	Disk Drive; 20.48-million words capacity; average access time 27 milliseconds; transfer rate 5.6 microseconds per word; controller select option for attachment to two controllers.	RHP04-E, RHP04-F, or RHP04-H	Note 3
RP04-C, RP04-C	Controller Select Option; converts RP04-A to RP04-B.	RP04-A	n/a
RP04-P, RP04-P	Disk Pack; additional disk pack for RP04 Disk Drive.	RP04-A or RP04-B	n/a
RHP04-AA, RHP04-AB	Disk System; includes controller and one RP04-A Disk Drive.	DF10 or DF10-C	4
RHP04-BA, RHP04-BB	Disk System; includes controller and one RP04-B Disk Drive.	DF10 or DF10-C	4
RHP04-DA, RHP04-DB	Disk System; includes DF10 Data Channel, controller and RP04-A Disk Drive.	MC10, MX10	4
RHP04-EA, RHP04-EB	Disk System; includes DF10 Data Channel, controller and one RP04-B Disk Drive.	MC10, MX10	4
RHP04-FA, RHP04-FB	Disk System; includes DF10-C Data Channel, controller and one RP04-A Disk Drive.	MC10, or MX10-C	4
RHP04-HA, RHP04-HB	Disk System; includes DF10-C Data Channel, controller and one RP04-B Disk Drive.	MC10, MX10-C	4
RTP04-AA, RTP04-AB	Disk System including Controller and RP04-A (single access) Disk Drive	KL10-B	4
RTP04-BA, RTP04-BB	Disk System including Controller and RP04-B (dual access) Disk Drive	KL10-B	4
RTP04-C	Dual Channel Access Kit	KL10-B, RTP04-A	N/A
RHP06-AA, RHP06-AB	Disk System; including Controller and RP06-A (single access) Disk Drive.	DF10 or DF10-C	4
RHP06-BA, RHP06-BB	Disk System; including Controller and RP06-B (dual access) Disk Drive.	DF10 or DF10-C	4
RP06-AA, RP06-AB	Add-on Disk Drive (single access); includes one RP06-P Disk Pack. 39, 40 million word capacity average access time. 27 milliseconds, transfer rate 5.6 microseconds per word.	RHP06-A or RHP06-B	32
RP06-BA, RP06-BB	Add-on Disk Drive (dual access); includes one RP06-P Disk Pack. 39, 40 million word capacity average access time, 27 milliseconds, transfer rate 5.6 microseconds per words.	RHP06-A or RHP06-B	32
RP06-C	RP06 Dual Access Kit	RP06-A	
RTP06-AA, RTP06-BB	Disk System including Controller and RP06-A (single access) Disk Drive	KL10-B	4
RTP06-BA, RTP06-BB	Disk System including Controller and RP06-B (dual access) Disk Drive	KL10-B	4
RTP06-C	Dual Channel Access Kit	KL10-B, RTP06-A	7
RHS04-DA, RHS04-DD	Swapping Disk; 256K words of swapping storage; transfer rate of four microseconds per word; room for one additional RHS04-C.	RHS04-G, RHS04-H or RHS04-J	16
RHS04-CA, RHS04-CD	Swapping Disk; 256K words of swapping storage; transfer rate four microseconds per word; for expansion of RHS04-D.	RHS04-D, RHS04-G RHS04-H, RHS04-J	16
RHS04-GA, RHS04-GD	Swapping System; includes controller and one RHS04-D Swapping Disk.	DF10 or DF10-C	2
RHS04-HA, RHS04-HD	Swapping System; includes DF10 Data Channel, controller, and one RHS04-D Swapping Disk.	MC10, MX10	2
RHS04-JA, RHS04-JD	Swapping System; includes DF10-C Data Channel, controller, and one RHS04-D Swapping Disk.	MC10 or MX10-C	2

OPTION DESIGNATOR 115V/60Hz 230V/50Hz	DESCRIPTION	PREREQUISITE	MAX. NO. SUP- PORTED (NOTE 1)
Magnetic Tape Systems			
TU10A-EE, TU10A-EJ	Magnetic Tape Drive; 9-channel drive, 45-inches-per-second; recording densities of 200, 556 and 88 bpi.	TM10-A or TM10-B	8
TU10A-FE, TU10A-FJ	Magnetic Tape Drive; 7-channel drive, 45-inches-per-second; recording densities of 200, 556 and 800 bpi.	TM10-A or TM10-B	8
TM10-AA, TM10-AB	Magnetic Tape Controller; Interfaces to I/O bus of KA10 or KI10 Central Processors for control of up to eight TU10A Magnetic Tape Drives.	KA10 or KI10	1
TU10-CA, TU10-CB	Magnetic Tape System; includes TM10-A Controller and one TU10A Magnetic Tape Drive.	KA10 or KI10	1
THU16-EA, THU16-ED	Magnetic Tape System; including RH10 and TM02 Controllers and TU16-E 9-track, 45 ips Tape Drive. Recording densities of 800/1600 bpi	DF10 or DAS33	8
TTU16-EA, TTU16-ED	Magnetic Tape System including Massbus Controller, Tape Controller and TU16-E Tape Drive	KL10-B	4
TU16B-E, TU16B-ED	Tape Controller and TU16-E Tape Drive (for expansion beyond 4 drives)	THU16-E or TTU16-E	8
TU16-EE, TU16-EJ	Add-on Magnetic Tape Drive, 9-track, 45 ips. Recording densities of 800/1600 bpi	THU16-E	32
TU40-A, TU40-B	Magnetic Tape Drive; 9-channel drive, 150-inches-per-second; recording densities of 200, 556 and 800 bpi.	TU40-C, TU41-C	16
TU41-A, TU41-B	Magnetic Tape Drive; 7-channel drive, 150-inches-per-second; recording densities 200, 556, and 800 bpi.	TU40-C, TU41-C	16
TM10-BA, TM10-BB	Magnetic Tape Control; attaches to DF10 or DF10-C Data Channel to control up to eight TU40 and/or TU41 Magnetic Tape Drives.	DF10 or DF10-C	2
TU40-CA, TU40-CB	Magnetic Tape System; includes DF10 Data Channel, TM10-B Controller and one TU40 Magnetic Tape Drive.	MC10, MX10	2
TU41-CA, TU41-CB	Magnetic Tape System; includes DF10 Data Channel TM10-B Controller and one TU41 Magnetic Tape Drive.	MC10, MX10	2
TU70-AA, TU70-AB	Magnetic Tape Drive; 9-channel drive, 200-inches-per-second; recording densities 800 and 1600 bpi.	TU70-C	16
TU71-AA, TU71-AB	Magnetic Tape Drive; 7-channel drive, 200-inches-per-second; recording densities 200, 556, and 800 bpi.	TU70-C	16
TU70-CA, TU70-CB	Magnetic Tape System; includes Data Channel, controller and one TU70-A Magnetic Tape Drive.	MC10, MX10 or MX10-C	2
TU56-A, TU56-B	DECTAPE Unit; dual-drive DECTAPE.	TD10 or TD10-G	8
TD10-A, TD10-B	DECTAPE Control; provides control for up to four TU56 DECTAPE units.	KA10	2
TD10-CA, TD10-CB	DECTAPE Control; provides control for up to four TU56 DECTAPE units.	KI10, KL10	2
TD10-GA, TD10-GB	DECTAPE System; includes TD10 Controller and one TU56 DECTAPE unit.	KI10 or KL10	2
TU72-C	Magnetic Tape System; includes DX10 data channel, TX02 tape control unit and TU72-E 9-track 125 ips Tape Drive. Recording densities of 1600 and 6250 bpi.	KI10 or KL10	2

OPTION DESIGNATOR 115V/60Hz 230V/50Hz	DESCRIPTION	PREREQUISITE	MAX. NO. SUP- PORTED (NOTE 1)
TU72-E	Magnetic Tape Drive; 9-track, 125-inches-per-second; recording densities 1600 and 6250 bpi.	TU72-C	8
Hard-copy Equipment			
CR10-DA, CR10-DB	Card Reader; table-top card reader, 1000-cards-per minute.	KA10, KI10 or KL10	2
CR10-EA, CR10-EB	Card Reader; console card reader, 1200-cards-per-minute.	KA10, KI10 or KL10	2
CR10-FA, CR10-FB	Card Reader; table-top card reader, 300-cards-per-minute.	KA10, KI10 or KL10	2
LSP10-LA, LSP10-LB	Line Printer; 245-lines-per-minute, 64-character print set.	KA10, KI10 or KL10	1, Note 5
LP10-FE, LP10-FE	Print Drum; 64-character EDP print drum for LP10-F Line Printer.	LP10-F	N/A
LP10-FF, LP10-FF	Print Drum; 64-character Scientific Print drum for LP10-F Line Printer.	LP10-F	N/A
LP10-HE, LP10-HE	Print Drum, 96-character EDP Print drum for LP10-H Line Printer.	LP10-H	N/A
LP10-HF, LP10-HF	Print Drum, 96-character Scientific Print drum for LP10-H Line Printer.	LP10-H	N/A
LP10-FA, LP10-FB	Line Printer; 1,250-lines-per-minute, 64-character EDP print set.	KA10, KI10, KL10	3
LP10-FC, LP10-FD	Line Printer; 1,250-lines-per-minute, 64-character Scientific print set.	KA10, KI10, KL10	3
LP10-HA, LP10-HB	Line Printer, 925-lines-per-minute, 96-character EDP print set.	KA10, KI10, KL10	3
LP10-HC, LP10-HD	Line Printer, 925-lines-per-minute, 96-character Scientific print set.	KA10, KI10, KL10	3
LP100-BA, LP100-BB	Line Printer; up to 1500LPM, EDP print set—see LP07-Y.	KA10, KI10, KL10	3
LP07-YA	Charaband for LP100 with 2 sets of 64-char. fonts.	LP100-B	N/A
LP07-YB	Charaband for LP100 with 2 sets of 96-char. fonts.	LP100-B	N/A
LP07-YC	Charaband for LP100 with 1 set of 64-char. fonts and 1 set of 96-char. fonts	LP100-B	N/A
XY10, XY10	Plotter Control; for CALCOMP Model, 563 or 565 plotter or equivalent.	KA10, KI10, KL10	2
XY10-A, XY10-A	Plotter; CALCOMP Model 565 plotter with control; uses 12-inch paper; specify step size: <div> <div>Step size</div> <div>Steps/minute</div> <div>0.01 inches</div> <div>18,000</div> <div>0.005 inches</div> <div>18,000</div> <div>0.1 inches</div> <div>18,000</div> </div>	KA10, KI10, KL10	2
XY10-B, XY10-B	Plotter; CALCOMP Model 563 plotter with control; uses 31-inch paper; specify step size: <div> <div>Step size</div> <div>Steps/minute</div> <div>0.01 inches</div> <div>12,000</div> <div>0.005 inches</div> <div>18,000</div> <div>0.1 inches</div> <div>18,000</div> </div>	KA10, KI10, KL10	2

Option Designator 115V/60Hz 230V/50Hz	DESCRIPTION	Prerequisite	Maximum No. Supported (Note 1)
Asynchronous Data Communication Systems			
DC10-AA, DC10-AB	Data Line Scanner; provides controller plus four units of cabinet space; DC10 System handles up to 64 lines per DC10-A.	KA10, KI10	2
DC10-B, DC10-B	Eight-line Group; Interface for eight local or dial-up lines; requires one unit of cabinet space; dial-up lines require DC10-E for full modem control.	DC10-A	16
DC10-C, DC10-C	Telegraph Relay Assembly; provides conversion from local to long lines using full- or half-duplex facilities; requires two units of cabinet space.	DC10-B and DC10-D	8
DC10-D, DC10-D	Telegraph Power supply; provides a standard line-voltage power supply used with DC10-C (120 volt, 2 amps); no cabinet space required.	DC10-C	8
DC10-E, DC10-E	Expanded Data Set Control; provides expanded control of eight data sets in the DC10; recommended for maximum system security; requires two units of cabinet space.	DC10-B	8
DC10-F, DC10-F	Expander Cabinet; provides eight units of cabinet space and power supplies for expansion beyond DC10-A.	DC10-A	2
DC10-H, DC10-H	Eight-line Group; interface for eight local or dial-up 2741-type lines; requires one unit of cabinet space; dial-up lines require DC10-E for full modem control.	DC10-A	8 Note 6
Remote Stations			
DAS80-AA, DAS80-AB	Remote Batch Station. Includes 1 DN81-H, LA36 Console, DAS80-CA/CB, DAS80-LA/LB, PDP-11 and software.	DAS 85, DN87, or DC75NP	N/A
DAS81-AA, DAS81-AB	Remote Concentrator. Includes PDP-11, 1 DN81-H, DN81-EA and software. Can add one more DN81-EC, ED for a total of 32 asynchronous lines. DN81-Fx asynchronous interfaces are also required and must be ordered separately.	DAS85, DN87, or DC75NP	N/A
DAS82-AA, DAS82-AB	Remote Batch Station and Concentrator. Includes PDP-11, DAS80-CA/CB card reader, DAS80-LA/LB line printer, 1 DN81-H, DN81-EA and software. DN81-Fx asynchronous interfaces are also required and must be ordered separately.	DAS85, DN87, or DC75NP	N/A
DAS92-AA, DAS92-AB	Includes 16K PDP-8/A processor, VT52, ROM for down-line loading, one synchronous line interface.	DECsystem-10 DC75-NP, DAS85 or DN87	N/A
DAS92-EA	Asynchronous 4 line multiplexer including line drivers. Will accommodate either 20 mA or EIA lines in any mixture.	See DAS92 configurator	4
DAS92-CA, DAS92-CB	Card Reader for 80-column (only) cards; operates at 285 cpm.	See DAS92 configurator	1
DAS92-PA, DAS92-PD	LA 180 Printer operates at 180 cps and uses a 96-character set.	See DAS92 configurator	1
DAS92-VA, DAS92-VD	LP05 Line Printer (64 characters) operates at 300 lpm; 132 columns wide carriage	See DAS92 configurator	1
DAS92-WA, DAS92-WD	LP05 Line Printer (96 characters).	See DAS92 configurator	1

Option Designator	DESCRIPTION	Prerequisite	Maximum No. Supported (Note 1)
Unit Record Options			
DAS80-CA, DAS80-CB	Card Reader, 300 cards per minute.	DAS80 Series Remote	1
DAS80-LA, DAS80-LB	Printer, 300 lines per minute, 132 printing positions, 64 printing characters, EDP character set.	DAS80 Series Remote	1
DAS80-LC, DAS80-LD	Printer, 230 lines per minute, 132 printing positions, 96 printing characters, EDP character set.	DAS80 Series Remote	1
Front Ends			
DN87-DA, DN87-DB	DN87 including DL10-C port only. Requires DL10-A with available port. Requires addition of DN81-xx synchronous and/or asynchronous line options.	KA10, KI10, KL10 DL10	6
DN87-AA, DN87-AB	DN87 including DL10-A Communications Interface and one DL10-C port. Requires addition of DN81-xx synchronous and/or asynchronous line options.	KA10, KI10, KL10	2
DN87S-AA, DN87S-AB	DN87 with DTE10 interface. Requires addition of DN81-xx synchronous and/or asynchronous line options.	1090	3
DN87-U	DC76 to DN87 software only upgrade (including KG11-A).	DC76	N/A
Asynchronous Line Options			
DN81-EA, DN81-EB	Asynchronous Expansion Cabinet including one DN81-EC 16-Line Asynchronous Expansion Group. Requires two DN81-Fx 8-Line Terminator Groups to activate the lines.	DN87, 87S*	2
DN81-EC, DN81-ED	Asynchronous 16-Line Expansion Group. This requires two DN81-Fx 8-Line Terminators to activate the lines.	DN87, 87S*	5
DN81-FA	8-Line Terminators each with 20 mA Current Loop Local Interfaces.	DN87, 87S	N/A
DN81-FB	8-Line Terminators each with EIA Local Interfaces.	DN87, 87S	N/A
DN81-FC	8-Line Terminators each with EIA Full Modem Control Interfaces.	DN87, 87S	N/A
DN81-FD	8-Line Terminators with Integral Auto Answer Modems. (These modems require customer supplied DAA's).	DN87, 87S	N/A
Synchronous Line Options			
DN81-EE, DN81-EF	Synchronous Expansion Cabinet. Includes one DN81-H.	DN87, 87S	1
DN81-H	Synchronous Single Line Controller Expansion Line. For data transmission speeds of up to 10K baud and for attachment to EIA RS232C compatible modems.	DN87, 87S	12
		DAS 80, 81, 82	4
DN81-J	Synchronous Single Line Controller. For data transmission speeds of up to 40.8K baud and for attachment to 303-type current mode modems.	DN87, 87S	12
		DAS 80, 81, 82	4

*Used on DN87, DN87S Front Ends and on DAS80 Series Remote Stations.

Option Designator 115V/60Hz 230V/50Hz		DESCRIPTION	Prerequisite	Maximum No. Supported (Note 1)
IBM DEVICE EMULATION AND TERMINATION FRONT ENDS				
DAS61-DA, DB	IBM 2780 and/or 3780 Emulation and Termination Front End. Including DL10-C port and requiring DL10-A with available port. Requires addition of DAS61-H and/or DAS61-J synchronous line units.	KI10, KL10, DL10 TOPS10 6.03	8	
DAS61-SA, SB	IBM 2780 and/or 3780 Emulation and Termination Front End including DTE10 interface. Requires addition of DAS61-H and/or DAS61-J synchronous line units.	KL10B	3	
DAS61-EA, EB	Synchronous Expansion Cabinet for expansion past 4 synchronous lines. Includes no synchronous lines.	DAS61-D, DAS61-S	1	
DAS61-H	Synchronous Single Line Controller for data transmission speeds of up to 10K baud and for attachment to EIA RS232C compatible modems.	DAS61-D, DAS61-S	12	
DAS61-J	Synchronous Single Line Controller for data transmission speeds of up to 50K baud and for attachment to 303-type current modems.	DAS61-D, DAS61S	12	
DAS61-QC	DAS61 Software Only	DAS61-D, DAS61-S	Note 6	
DAS61-QX	DAS78 to DAS61 Software Only Upgrade (prerequisite GALAXY).	DAS61-D, DAS61-S	Note 6	
DAS62-DA, DB	IBM 2780 and/or 3780 and/or HASP multi-leaving work station Front End. Including DL10-C port and requires DL10-A with available port. Requires addition of DAS62-H and/or DAS62-J synchronous line units.	KI10, KL10, DL10 TOPS10 6.03	4	
DAS62-SA, SB	IBM 2780 and/or 3780 and/or HASP multi-leaving work station Front End including DTE10 interface. Requires addition of DAS62-H and/or DAS62-J synchronous line units.	KL10B	8	
DAS62-EA, EB	Synchronous Expansion Cabinet for expansion past 4 synchronous lines. Includes no synchronous lines.	DAS62-D, DAS62-S	3	
DAS62-H	Synchronous Single Line Controller for data transmission speeds of up to 10K baud and for attachment to EIA RS232C compatible modems.	DAS62-D, DAS62-S	12	
DAS62-J	Synchronous Single Line Controller for data transmission speeds of up to 50K baud and for attachment to 303-type current modems.	DAS62-D, DAS62-S	12	
DAS62-QC	DAS62 Software Only.	DAS62-D, DAS62-S	Note 6	
DAS62-QX	DAS78 to DAS62 Software Only Upgrade (prerequisite GALAXY).	DAS78	Note 6	
DAS62-QY	DAS61 to DAS62 Software Only Upgrade.	DAS61	Note 6	
Terminals				
LA36-CC, LA36-CD	DECwriter-II; 30-character-per-second teleprinter with 20-mA current loop interface; handles up to 6-part forms, 132-column printing, variable-width forms, 96 upper- and lower-case characters.	Asynchronous Line	Note 4	
LA37	APL-ASCII Terminal, 30-character-per-second, features full APL, full upper/lower case, or upper case TTY character sets.	"	Note 4	
VT05B-DA, VT05-DB	Alphanumeric Terminal; 110 to 2400-baud CRT display terminal; features 128-character keyboard, 1440-character capacity screen, direct cursor addressing, 20-mA current loop or EIA interface.	"	Note 4	

Option Designator 115V/60Hz 230V/50Hz	DESCRIPTION	Prerequisite	Maximum No. Supported (Note 1)
GT40-AA, GT40-AB	Graphic System; computer-level graphic system consisting of a graphical display system and general-purpose minicomputer; features light pen, 128-character keyboard and EIA serial interface.	"	Note 4
VT50-CA, VT50-CD	DECscope; 75 to 9600-baud CRT terminal, 20-mA current loop interface, 960-character capacity screen, 64-character keyboard.	"	Note 4
VT52	DECscope; 75 to 9600-baud CRT terminal, EIA interface, 1920-character capacity screen, 96-character keyboard.	"	Note 4
VT50-EA, VT50-ED	DECscope; 75 to 9600-baud CRT terminal, EIA interface, 960-character screen, 64-character keyboard.	"	Note 4
VT61	DECscope Video Display Block-mode Terminal; 1920-character display and full ASCII keyboard; 128-character 7x8 dot matrix character set.	"	Note 4 Note 6
Miscellaneous			
DA10, DA10	Interface between KA10 or KI10 CPU and PDP-8 minicomputer.	KA10, KI10, KL10	n/a
DK10, DK10	Real-time Clock; provides resolution to 10 micro-seconds via crystal oscillator.	KA10, or KI10	2
PC10-C	High speed paper tape reader/punch	KL10	1
GP10, GP10	General Purpose Interface; provides interface to KA10 or KI10 I/O bus; includes cabinet, two model 728 power supplies, one model 844 power control indicator in panels, convenience outlet and BS10-A-15 cable set; logic provides a status register, device decoding, read-in gating and line buffering.	KA10, KI10, KL10	N/A
GP10-L, GP10-L	GP10 Interface Logic; interface logic only; does not include power supply or indicators.	KA10, KI10, KL10	N/A
GP10-M, GP10-M	GP10 less logic; includes power supplies and indicators; does not include cables.	KA10, KI10, KL10	N/A
DF01-AA, DF01-AB	Acoustic Telephone Coupler.	DECsystem-10 Terminal	Note 4
CAB-9B, CAB-9B	Cabinet; includes full-length single doors, front and back with indicator panel, but without end panels (old style).	DECsystem-10	N/A
H-956, H-956	Cabinet; includes full-length single doors, front and back with indicator panel, but without end panels (new style).	DECsystem-10	N/A

Note: Specifications subject to change without notice.

Note 1: Maximum number supported by standard software. Configurations with large numbers of disk drives, high-speed magnetic tape drives and/or data communication lines must be reviewed by DECsystem-10 Marketing.

Note 2: Maximum memory capacity of KA10 is 256K words in 16 memory modules. Maximum memory capacity of KI10 or KL10 is 4,096K words in 16 memory modules.

Note 3: Currently under software development. Contact DECsystem-10 Marketing before configuring.

Note 4: DECsystem-10's support a maximum of 512 simultaneously-active terminals through 127 simultaneously-active jobs.

Note 5: Software Supported by Computer Special Systems.

Note 6: Software Supported by Advanced Systems Group.

DECsystem-10 UPGRADE/MIGRATION PATHS

This chart illustrates the migration of different DECsystem-10 subsystems toward the common TOPS-10/TOPS-20 hardware system. The chart is useful for determining what equipment is supported within each DECsystem-10 family member. It also provides insight into what equipment needs to be upgraded in moving from one CPU-based system to another.

SYSTEM DESIGNATION	1040 1050 1055	1060 1070 1077	1080	1090	1090 EXTENDED FULLY COMPATIBLE HARDWARE FOR TOPS-10 OR TOPS-20
OPERATING SYSTEM	TOPS-10	TOPS-10	TOPS-10	TOPS-10	TOPS-10 ①→TOPS-20
CPU	KA10 ————— ②	KI10 ————— ③	KL10 ————— ④	KL10B ————— ⑤	KL10B EXTENDED
MEMORY	MA10 —————→ MA10* ME10 (16K) —————→ ME10 MF10 (32K) —————→ MF10 MG10 (128K) —————→ MG10 MH10 (256K) —————→ MH10			ME10 ————— ② MF10 —————→ MF10 MG10 —————→ MG10 MH10 —————→ MH10	
DISK	RP10/RP02-3 —————→ RP10/RP02-3 RH10/RP04 & 06 —————→ RH10/RP04 & 06 RC10/RM-RD10 —————→ RC10/RM-RD10 ②		RP10/RP02-3* ————— ③ RH10/RP04 & 06 ————— ④ RH10/RS04 ————— ⑤	RH10/RP04 & 06 ————— ④ RH20/RP04 & 06 ————— ⑤	RH20/RP04 & 06
COMMUNICATIONS	DC10 —————→ DC10 DC76 —————→ DC76 DAS75 ————— ②	DC10 —————→ DC10 DAS85 ————— ②	DC10* ————— ③ DC76 ————— ④ DN87 ————— ⑤	DN87 ————— ⑤ DN87S ————— ⑥	DN87S
MAGTAPE	TM10A/TUXX —————→ TM10A/TUXX ⑥ TM10/TUXX —————→ TM10B/TUXX ⑦ RH10/TU16 —————→ RH10/TU16 ④ DX10/TU7X —————→ DX10/TU7X ⑦ TC10/TSU43 —————→ TC10/TSU43 ②		TM10B/TUXX ————— ⑦ RH10/TU16 ————— ④ DX10/TU7X ————— ⑦	RH20/TU16 ————— ⑧ DX10/TU7X ————— ⑦ DX20/TUXX ————— ⑨	RH20/TU16 DX20/TUXX
UNIT RECORD	BA10/DEVICE —————→ BA10/DEVICE	BA10/DEVICE —————→ BA10/DEVICE	BA10/DEVICE —————→ BA10/DEVICE	BA10/DEVICE ————— ⑧	PDP-11 FRONT END DEVICES

NOTES:

- 1 New microcode; TOPS-20 Release.
- 2 Drop-in upgrade replacement (old component typically returned/retired)
- 3 Field retrofit (in-place)
- 4 Substitution of RH20 for DF10/RH10
- 5 Field Upgrade: replace DL10 with DTE connection, S/W Release
- 6 Field Upgrade/Addition: replace/upgrade TM10A with TM10B and add DF10/DF10-C
- 7 Tape channel replaced with DX20 (internal channel)
- 8 BA10 Hard Copy Control Unit replaced with PDP-11-based controller(s)
- * Starred items are those hardware components which are not manufactured, difficult/impossible to support, or whose successor provides cost/performance or other features which eventually make their support uneconomical.

SOFTWARE

SOFTWARE	59
Languages	61
AID	60
ALGOL	60
APL	60
BASIC	61
COBOL	62
ISAM	62
CPL	62
FORTRAN	63
FORTRAN Version 4	63
MACRO	64
System Utilities	65
SORT	65
Data Base Management System	65
ITPS-10	66
LINK-10	66
MACY11-LNKX11	67
DDT	67
FAILSAFE/BACKUP	67
Message Control System	68
PIP	68
RUNOFF	68
TECO	68
Operating System	69
TOPS-10 Operating System	69
Command Control Language	72
File System	73
Input/Output	76
Memory Management	76
Multi-Processor Systems	77
Inter-job Communication	78
Communications Software	78
Multimode Computing	79
Timesharing	79
GALAXY-10 Batch	79
Real-Time Computing	81
Diagnostic Software	83
High Availability	83
Maintenance Features	83
On-Line Diagnosis	83

Languages

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-10 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.

AID

The Algebraic Interpretive Dialogue, AID, is the DECsystem-10 adaptation of the language elements of JOSS, a program developed by the RAND Corporation. To write a program in the AID language requires no previous programming experience. Commands to AID are typed in via the user's terminal as imperative English sentences. Each command occupies one line and can be executed immediately or stored as part of a routine for later execution. The beginning of each command is a verb taken from the set of AID verbs. These verbs allow the user to read, store and delete items in storage; halt the current processing and either resume or cancel execution; type information on his terminal; and define arithmetic formulas and functions for repetitive use that are not provided for in the language. However, many common algebraic and geometric functions are provided for the user's convenience.

The AID program is device-independent. The user can create external files for storage of subroutines and data for subsequent recall and use. These files may be stored on any retrievable storage media, but for accessibility and speed, most files are stored on disk.

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

BASIC

- Multiple users share the BASIC compiler's pure code.
- A variety of program manipulation commands which include functions for savings, 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 facilities. This file may now be referenced and written either sequentially or randomly.

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

DECsystem-10 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-10 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-10 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-10 operating system.

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 time-sharing, 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, 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 for 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

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 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 Version 4

FORTRAN-10 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-10:

- **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 primary source file
- **GLOBAL OPTIMIZATION-FORTRAN-10** performs extensive local and optional global optimizations
- **FORDDDT**—the FORTRAN-10 debugger

FORDDDT in conjunction with the FORTRAN-10/"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 logical functions (rather than a result of just true or false)
- **END=** and **ERR=** in I/O statements
- Device independence

• Data Base Management System Interface

The programmer may call upon the data base management system facilities from within his program. This system, which allows 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.

The FORTRAN-10 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-10 is easy to use in both timesharing and batch processing environments. Under timesharing, the user operates in an interactive editing and debugging environment. FORDDDT, an interactive program that is used as an aid in debugging.

FORTRAN programs use 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 program on the DECsystem-10. It makes machine language programming easier and faster for the user by

- Translating symbolic operation codes in the source program into the binary codes needed in machine language instructions.
- Relating symbols specified by the user to numeric values.
- Assigning absolute core addresses to the symbolic addresses of program instructions and data.
- Preparing an output listing of the program which includes any errors detected during the assembly process.

MACRO programs consist of a series of statements that are usually prepared on the user's terminal with a system editing program. The elements in each statement do not have to be placed in certain columns nor must they be separated in a rigid fashion. The assembler interprets and processes these statements, generates binary instructions or data words, and performs the assembly.

MACRO is a two-pass assembler. This means that the assembler reads the source program twice. Basically, on the first pass, all symbols are defined and placed in the symbol table with their numeric values, and on the second pass, the binary (machine) code is generated. Although not as fast as a one-pass assembler, MACRO is more efficient in that less core is used in generating the machine language code and the output to the user is not as long.

MACRO is a device-independent program; it allows the user to select, at runtime, standard peripheral devices for input and output files. For example, input of the source program can come from the user's terminal, 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 correct sequence. This capability allows for the expansion and adaptation of the assembler in order to perform specialized functions for each programming job.

System Utilities

SORT

The TOPS-10 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.

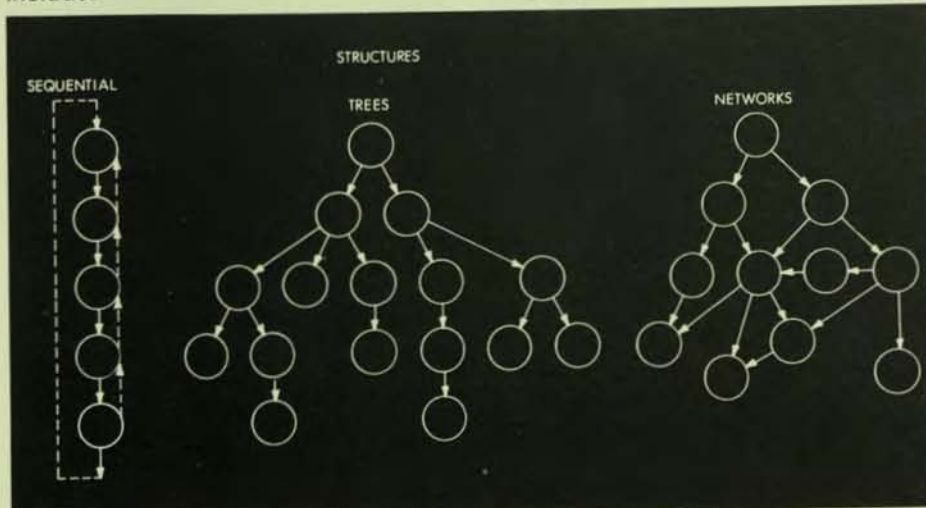
DIGITAL offers a solution to the problem in the Data Base Management System—DBMS. It enables DEC system-10 users to organize and maintain data in forms more suitable to the integration of a number of related but separate processes and applications. DBMS is ideal in situations where data processing control and program development functions require structures and techniques not satisfied by traditional data management facilities.

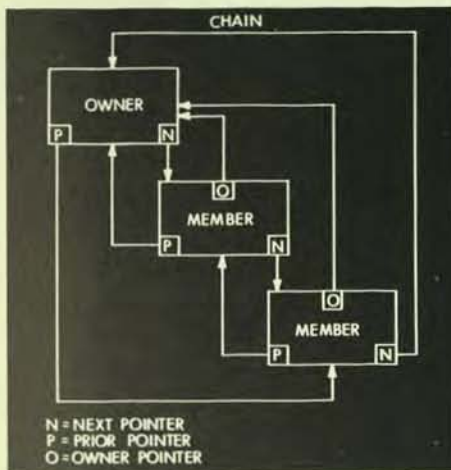
Features

The DECsystem-10 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-10 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-10 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-data 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 re-compilations.
- **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 the 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

ITPS-10

ITPS-10, In-House Text Preparation System allows DECsystem-10 users to prepare office-quality or camera-ready (typeset) copy using the interactive capability of the DECsystem-10. High-quality, camera-ready copy for pamphlets, advertising brochures, product flyers, user manuals, and many other types of documentation can be produced and updated rapidly and inexpensively using ITPS-10. The system allows use of many different input and output devices, including interactive CRT or hard copy terminals, line printers, typewriterlike devices, and photocomposition equipment. Documents are stored as standard DECsystem-10 files and can be easily updated, edited and re-generated.

The system is easy-to-use and easy to learn. Current successful applications range from preparing form letters to setting daily newspapers.

LINK-10

LINK-10, the DECsystem-10 linking loader, merges independently-translated modules of the user's program into a single module and links this module with system modules into a form that can be executed by the operating system. It provides automatic relocation and loading of the binary modules producing an executable version of the user's program. When the loading process has been completed, the user can request LINK-10 either to transfer control to his program for immediate execution or to output the program to a device for storage in order to avoid the loading procedure in the future.

While the primary output of LINK-10 is the executable version of the user's program, the user can request auxiliary

output in the form of map, log, save, symbol, overlay plot, and expanded core image files. This additional output is not automatically generated: the user must include appropriate switches in his command strings to LINK-10 in order to obtain this type of output. 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 switches in his command strings to LINK-10, the user can 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 verbosity levels of the messages. Finally, he can accept the LINK-10 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.

LINK-10 has an overlay facility to be used when the total core required by a user's program is more than the core available to the user. The user organizes his program so that only some portions of the program are required in core at any one time. The remaining portions reside in a disk file and are transferred in and out of core during execution. Because the portion brought into core may overlay a portion already core-resident, the amount of core required by the entire program is reduced. Overlays can be invoked either by runtime routines called from the user's program or by automatic calls to subroutines outside the current overlay link.

MACY11 - LNKX11

MACY11 is a cross-assembler which operates on the DECsystem-10 and assembles MACRO-11 source code for the PDP-11 family of computers.

MACY11 will produce either an absolute binary or a relocatable object module, depending upon the assembly mode. The resulting absolute binary files are transferrable to the PDP-11 for execution or, the relocatable object modules may be used as input to the LNKX11 linker on the DECsystem-10 and then transferred to the PDP-11 operating environment. MACY11 will append a symbol table and (optionally) a cross-reference table to the listing file. MACY11 produces a "side-by-side" assembly listing of symbolic source statements, their octal equivalents, assigned addresses and error codes.

DDT

The Dynamic Debugging Technique, DDT, is used for on-line program composition of object programs and for on-line checkout and testing of these programs. For example, the user can perform rapid checkout of a new program by making a change resulting from an error detected by DDT and then immediately executing that section of the program for testing.

After the source program has been compiled or assembled, the binary object program with its table of defined symbols is loaded with DDT. In command strings to DDT, the user can specify locations in his program, or 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 any location. Insertions and deletions can be in source language code or in various numeric and text modes. DDT also performs searches, gives conditional dumps, and calls user-coded debugging subroutines at breakpoint locations.

FAILSAFE/BACKUP

The FAILSAFE/BACKUP programs are used to selectively or universally save disk files on magnetic tape. Files may be copied to magnetic tape, in a special format, selectively by file name or groups of file names, by project/programmer number, by disk structure (logical or physical), or universally. In addition to providing backup of files, these programs allow a method of extending disk space by allowing infrequently-used files to be stored on tape. Restoring of files from FAILSAFE/BACKUP format tapes can likewise be handled on an individual file, groups of files, project/programmer, logical or physical disk basis. An added benefit is achieved when restoring files in that disk fragmentation is often minimized by the failsafe/restore operation.

Message Control System

The Message Control System (MCS-10) is to provide the DECsystem-10 with an efficient, transaction processing-oriented system that facilitates control of communications between a network of terminals and applications processes. The intent is to provide generalized routines that are readily used from COBOL programs and network definitions. Message control in its fullest sense includes a complex of hardware and software systems:

- Terminals and transmission devices.
- Remote and central site concentrators.
- Central site communications multiplexers.
- DECsystem-10 Monitor and its Message Service module.
- Message Control System (MCS-10) Routines.
- Extended Communications Language module and interfaces (COBOL/LIBOL).
- Network Definition, Activation and Operations Control Routines.
- Message Oriented Applications Programs.
- Data Base Management capability via DBMS-10.

The activities normally associated with message control are cooperative processes distributed among communications concentrators, multiplexers, and a central system. Actual I/O transmission, line control, error handling, device selection, code translation, and message buffering are handled by the remote and local concentrators and front-end processors. The DECsystem-10 central processor, Monitor, and Message Control modules accept, route, queue and log terminal and application program-generated messages. Upon the occurrence of specific user-specified events, application programs are activated to perform message processing. Applications programs receive messages from queues through interface modules, perform processes related to data within messages and send messages back through queues to terminals and/or other application processes.

PIP

The Peripheral Interchange Program, PIP, is used to transfer data files from one I/O device to another. Commands to PIP are formatted to accept any number of input (source) devices and one output (destination) device. Files can be transferred from one or more source devices to the destination device as either one combined file or individual files. Switches contained in the command string to PIP provides the user with the following capabilities:

- Naming the files to be transferred.
- Editing data in any of the input files.
- Defining the mode of transfer.
- Manipulating the directory of a device if it has a directory.
- Controlling magnetic tape and card punch functions.
- Recovering from errors during processing.

RUNOFF

RUNOFF facilitates the preparation of typed or printed manuscripts by performing line justification, page numbering, titling, indexing, formatting, and case shifting as directed by the user. The user creates a file with an editor and enters his material through his terminal. In addition to entering the text, the user includes information for formatting and case shifting. 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 the text that will remain unchanged. After the group of modifications have been added to the file, RUNOFF produces a new copy of the file which is properly paged and formatted.

TECO

The Text Editor and COrrector program, TECO, is a powerful editor used to edit any ASCII text file with a minimum of effort. TECO commands can be separated into two groups: one group of elementary commands that can be applied to most editing tasks, and the larger set of sophisticated commands for character string searching, text block movement, conditional commands, programmed editing, and command repetition.

TECO is a character-oriented editor. This means that one or more characters in a line can be changed without retyping the remainder of the line. TECO has the capability to edit any source document: programs written in MACRO, FORTRAN, COBOL, ALGOL, or any other source language; specifications; memoranda, and other types of arbitrarily-formatted text. The TECO program does not require that line numbers or other special formatting be associated with the text.

Editing is performed by TECO via an editing buffer, which is a section within TECO's core area. Editing is accomplished by reading text from any device into the editing buffer (inputting), by modifying the text in the buffer with data received from either the user's terminal or some other device (inserting), and by writing the modified text in the buffer to an output file (outputting).

A position indicator, or buffer pointer, is used to locate characters within the buffer and its position determines the effect of many of TECO's commands. It is always positioned before the first character, between two characters, or after the last character in the buffer. Various commands, such as insertion commands, always take place at the current position of the buffer pointer.

There are TECO commands to manipulate data within the editing buffer. Input and output commands read data from the input file into the buffer and output data from the buffer to the output file. There are other commands to have one or more characters inserted into the editing buffer, deleted from the buffer, searched for, and/or typed out. In addition, the user can employ iteration commands to execute a sequence of commands repeatedly and conditional execution commands to create conditional branches and skips.

Operating System

TOPS-10 Operating System

The resident operating system is made up of a number of separate and somewhat independent parts, or routines. Some of these routines are cyclic in nature and are repeated at every system clock interrupt (tick) to ensure that every user of the computing system is receiving the requested services. These cyclic routines are

- The command processor, or decoder
- The scheduler
- The swapper

The command decoder is responsible for interpreting commands typed by the user on his terminal and passing them to the appropriate system program or routine. The scheduler decides which user is to run in the interval between the clock interrupts, allocates sharable system resources, and saves and restores conditions needed to start a program interrupted by the clock. The swapper rotates user jobs between secondary memory (usually disk or drum) and core memory after deciding which jobs should be in core but are not. These routines constitute the part of the operating system that allows many jobs to be operating simultaneously.

The non-cyclic routines of the operating system are invoked only by user programs and are responsible for providing these programs with the services available through the operating system. These routines are

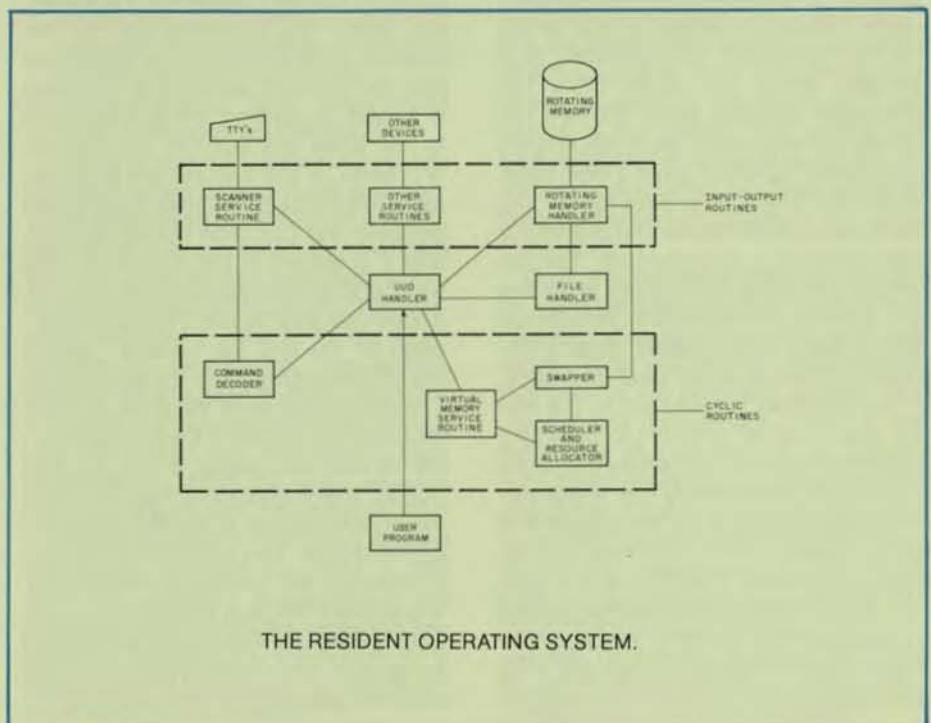
- The UUO handler
- The input/output routines
- The file handler

The UUO handler is the means by which the user program communicates with the operating system in order to have a service performed. Communication is by way of programmed operators (also known as UUO's) contained in the user program which, when ex-

ecuted, go to the operating system for processing. The input/output routines are the routines responsible for directing data transfers between peripheral devices and user programs in core memory. These routines are invoked through the UUO handler, thus saving the user the detailed programming needed to control peripheral devices. The file handler adds permanent user storage to the computing system by allowing users to store named programs and data as files.

Scheduler

The DECsystem-10 is a multiprogramming system; i.e., it allows several user jobs to reside in core simultaneously and to operate sequentially. It is then the job of the scheduler to decide which jobs should run at any given time. In addition to the multiprogramming feature, the DECsystem-10 employs a swapping technique whereby jobs can exist on an external storage device (e.g., disk or drum) as well as in core. Therefore, the scheduler decides not only what job is to be run next but also when a job is to be swapped out onto disk or drum and later brought back into core.



All jobs in the system are retained in ordered groupings called queues. These queues have various priorities that reflect the status of each job at any given moment. The queue in which a job is placed depends on the system resource for which it is waiting and, because a job can wait for only one resource at a time, it can be in only one queue at a time. Several of the possible queues in the system are

- Run queues for jobs waiting for, or jobs in, execution.
- I/O wait queues for jobs waiting for data transfers to be completed.
- Resource wait queues for jobs waiting for some system resource.
- Null queue for all job numbers that are not currently being used.

The job's position within certain queues determines the priority of the job with respect to other jobs in the same queue. For example, if a job is first in the queue for a sharable device, it has the highest priority for the device when it becomes available. However, if a job is in an I/O wait queue, it remains in the queue until the I/O is completed. Therefore, in an I/O wait queue, the job's position has no significance. The status of a job is changed each time it is placed into a different queue.

The scheduling of jobs into different queues is governed by the system clock. This clock divides the time for the central processor into sixtieths of a second. When the clock ticks, the scheduler decides which job will run during the next cycle. Each job, when it is assigned to run, is given a time slice. When the time slice expires for the job, the clock notifies the central processor and scheduling is performed. The job whose time slice just expired is moved into another, perhaps lower, priority run queue, and the scheduler selects another job to run in the next time slice. If the currently running job is the null job and a higher priority job (any job) becomes ready to run before the clock ticks it will immediately run. If the currently running job is not the null job and a high priority (HPQ real-time) job becomes runnable it will run at the next clock tick. Finally if a job just becoming runnable is not an HPQ job, but is of higher priority than the current job, it will only preempt the current job when the applicable time slice has expired.

Each job, when it is assigned to run, is given a time slice. When the time slice expires for the job, the clock notifies the central processor and scheduling is performed. The job whose time slice just expired is moved into another, perhaps lower, priority run queue, and the scheduler selects another job to run in the next time slice.

Scheduling may be forced before the clock ticks if the currently-running job reaches a point at which it cannot immediately continue. Whenever an operating system routine discovers that it cannot complete a function requested by the job (e.g., it is waiting for I/O to complete or the job needs a device which it currently does not have), it calls the scheduler so that another job can be selected to run. The job that was stopped is then requeued and is scheduled to be run when the function it requested can be completed. For example: when the currently running job begins input from a DECtape, it is placed into the I/O wait queue, and the input is begun. A second job is scheduled to run while the input of the first job proceeds. If the second job then decides to access a DECtape, it is stopped because the DECtape control is busy, and it is placed in the queue for jobs waiting to access the DECtape control. A third job is set to run. The input operation of the first job finishes, freeing the DECtape control for the second job. The I/O operation of the second job is initiated, and the job is transferred from the device wait queue to the I/O wait queue. The first job is transferred from the I/O wait queue to the highest-priority run queue. This permits the first job to preempt the running of the third job. When the time slice of the first job becomes zero, it is moved into the second run queue, and the third job runs again until the second job completes its I/O operation.

In addition, data transfers allow the scheduler to permit the user to overlap computation with data transmission. In unbuffered data modes, the user supplies an address of a command list containing pointers to locations in his area to and from which data is to be transferred. When the transfer is initiated, the job is scheduled into an I/O wait queue where it remains until the device signals the scheduler that the entire transfer has been completed.

In buffered modes, each buffer contains information to prevent the user and the device from using the same buffer at the same time. If the user requires the buffer currently being used by the device as his next buffer, the user's job is scheduled into an I/O wait queue. When the device finishes using the buffer, the device calls the scheduler to reactivate the job.

TOPS-10 also provides the ability to tune the system scheduler to specific loads. The system administrator can set aside percentages of the CPU to given classes of timesharing and batch users. This facility can be dynamically changed while the system is running and need not be defined at system generator time, thus facilitating optimal system usage over a wide variety of system loads.

Swapper

The swapper is responsible for keeping in core the jobs most likely to be run. It determines if a job should be in core by scanning the various queues in which a job may be. If the swapper decides that a job should be brought into core, it may have to take another job already in core and transfer it to secondary storage. Therefore, the swapper is not only responsible for bringing a job into core but is also responsible for selecting the job to be swapped out.

A job is swapped to secondary storage for one of two reasons:

- A job that is more eligible to run needs to be swapped in and there is not enough room in core for both jobs.
- The job needs to expand its core size and there is not enough core space to do so.

If the latter case is true, the job must be swapped out and then swapped in later with new allocation of core.

The swapper checks periodically to see if a job should be swapped in. If there is no such job, then it checks to see if a job is requesting more core. If there is no job wishing to expand its size, then the swapper does nothing further and relinquishes control of the processor until the next clock tick.

UUO Handler

The UUO handler is responsible for accepting requests for services available through the operating system. These requests are made by the user program via software-implemented instructions known as programmed operators, or UUO's. The various services obtainable by the user program include:

- Communicating with the I/O devices on the computing system, including connecting and responding to any special devices that may be desired on the system for real-time programming.
- Receiving or changing information concerning either the computing system as a whole or the individual program.
- Altering the operation of the computing system as it concerns the user job, such as controlling execution by trapping or suspending, or controlling core memory by locking.
- Communicating and transferring control between user programs.

The UUO handler is the only means by which a user program can give control to the operating system in order to have a service performed. Contained in the user program are operation codes which, when executed, cause the hardware to transfer control to the UUO handler for processing. The routine obtains its arguments from the user program. The core location at which the UUO operation was executed is then remembered. After the UUO request has been processed, control is returned to the user program at the first or second instruction following the UUO. In this way, the software supplements the hardware by providing services that are invoked through the execution of a single core location just as the hardware services are invoked.

Device Service Routines

I/O programming in the DECsystem-10 is highly convenient for the user because all of the burdensome details of programming are performed by the operating system. The user informs the operating system of his requirements for I/O by means of UUO's contained in his program. The actual input/output routines needed are then called by the UUO handler.

Since the operating system channels communication between the user program and the device, the user does not need to know all the peculiarities of each device on the system. In fact, the user program can be written in a similar manner for all devices. The operating system will ignore, without returning an error message, operations that are not pertinent to the device being used. Thus, a terminal and a disk file can be processed identically by the user program. In addition, user programs can be written to be independent of any particular device. The operating system allows the user program to specify a logical device name, which can be associated with any physical device at the time when the program is to be executed. Because of this feature, a program that is coded to use a specific device does not need to be rewritten if the device is unavailable. The device can be designated as a logical device name and assigned to an available physical device with one command to the operating system.

Data is transmitted between the device and the user program in one of two methods: unbuffered mode or buffered mode. With unbuffered data modes, the user in his program supplies the device with an address, which is the beginning of a command list. Essentially, this command list contains pointers specifying areas in the user's allocated core to or from which data is to be transferred. The user program then waits until the operating system signals that the entire command list has been processed. Therefore, during the data transfer, the user program is idly waiting for the transfer to be completed.

Data transfers in buffered mode utilize a ring of buffers set up in the user's core area. Buffered transfers allow the user program and the operating system's I/O routines to operate asynchronously. As the user program uses one buffer, the operating system processes another one by filling or emptying it as interrupts occur from the device. To prevent the user program and the operating system from using the same buffer at the same time, each buffer has a use bit that designates who is using the buffer. Buffered data transfers are more efficient than unbuffered transfers because the user program and the operating system can be working together in processing the data.

Several steps must be followed by the user program in order for the operating system to have the information it needs to control the data transfers. Each step is indicated to the operating system with one programmed operator. In the first step, the specific device to be used in the data transfer must be selected and linked to the user program with one of the software I/O channels available to the user's job (OPEN or INIT programmed operators). This device remains associated with the software I/O channel until it is disassociated from it (via a programmed operator) or a second device is associated with the same channel. In addition to specifying the I/O channel and the device name, the user program can supply an initial file status, which includes the

type of data transfer to be used with the device (e.g., ASCII, binary), and the location of the headers to be used in buffered data transfers. The operating system stores information in these headers when the user program executes programmed operators, and the user program obtains from these headers all the information needed to fill or empty buffers.

Another set of programmed operators (INBUF and OUTBUF) establishes the actual buffers to be used for input and output. This procedure is not necessary if the user is satisfied to accept the two buffers automatically set up for him by the operating system.

The next step is to select the file that the user program will be using when reading or writing data. This group of operators (LOOKUP and ENTER) is not required for devices that are not file-structured (e.g., card reader, magnetic tape, paper-tape punch). However, if used, they will be ignored, thus allowing file-structured devices to be substituted for non-file-structured devices without the user rewriting the program.

The third step is to perform the data transmission between the user program and the file (IN, INPUT, OUT, and OUTPUT). When the data has been transmitted to either the user program on input or the file on output, the file must be closed (CLOSE, fourth step) and the device released from the channel (RELEASE, fifth step). This same sequence of programmed operators is performed for all devices; therefore, the I/O system is truly device-independent because the user program does not have to be changed every time a different device is used.

In addition to reading or writing data to the standard I/O devices, provisions are included in the operating system for using the terminal for I/O during the execution of the user program. This capability is also obtained through programmed operators. As the user program is running, it can pause to accept input from or to type output to the terminal. The operating system does all buffering for the user, thus saving programming time. This method of terminal I/O provides the user with a convenient way of interacting with a running program.

Command Control Language

By allowing resources to be shared among users, the timesharing environment utilizes processor time and system resources that are wasted in single-user systems. Users are not restricted to a small set of system resources, but instead are provided with the full variety of facilities. By means of his terminal, the user has on-line access to most of the system's features. This on-line access is available through the operating system command control language, which is the means by which the timesharing user communicates with the operating system.

Through the command language, the user controls the running of a task, or job, to achieve the desired results: create, edit, and delete files; start, suspend, and terminate a job; compile, execute, and debug a program. In addition, since GALAXY batch software accepts the same command language as the timesharing software, any user can enter a program into the batch run queue. Thus, any timesharing terminal can act as a remote job entry terminal.

When the user types commands and/or requests on his terminal, the characters are stored in an input buffer in the operating system. The command decoder examines these characters in the buffer, checks them for correct syntax, and invokes the system program or user program as specified by the command.

File System

Mass storage devices, such as disks and drums, cannot be requested for a user's exclusive use, but must be shared among all users. (An exception is the assignment of a disk structure to a single user; i.e., a "private" disk structure.) Because many users share these devices, the operating system must ensure independence among the users; one user's actions must not affect the activities of another unless the users desire to work together. To guarantee such independence, the operating system provides a file system for disks, disk packs, and drums. Each user's data is organized into groups of 128-word blocks called files. The user gives a name to each of his files, and the list of these names is kept by the operating system for each user. The operating system is then responsible for protecting each user's file storage from intrusion by unauthorized users.

In addition to allowing independent file storage for users the operating system permits sharing of files among individual users. For example, programmers working on the same project can share the same data in order to complete a project without duplication of effort. The operating system lets the user specify protection rights, or codes, for his files. These codes designate if other users may read the file, and after access, if the files can be modified in any way. A new facility called File DAEMON allows the user to specifically permit or deny access to any file, or set of files, for specific users. The user may also create a log of accesses to his files for later review, proprietary program billing, etc.


The user of the DECsystem-10 is not required to preallocate file storage; the operating system allocates and deallocates the file storage space dynamically on demand. Not only is this convenient for the user because he does not have to worry about allocation when creating files, but this feature also conserves storage by preventing large portions of storage from being unnecessarily tied up. However, a large batch job which needs to preallocate space may do so.

Files are assigned protection levels for each of three classes of users: self; users with a common project number; and all users. Each user class may be assigned a different access privilege; there are eight levels in each of the three user classes. The owner of a file may always change its protection.

In addition the File DAEMON is called on all access failures if the owner protection is 4,5,6,7 . . . Note that the protection used by the system to determine access right after a File DAEMON call is provided by the file DAEMON.

On each clock interrupt, control is given to the command decoder to interpret and process one command in the input buffer. The command appearing in the input buffer is matched with the table of valid commands accepted by the operating system. A match occurs if the command typed in exactly matches a command stored in the system, or if the characters typed in match the beginning characters of only one command (i.e., constitute a unique abbreviation). When the match is successful, the legality information (or flags) associated with the command is checked to see if the command can be performed immediately. For instance, a command must be delayed if the job is swapped out to the disk and command requires that the job be resident in core; the command is executed on a later clock interrupt when the job is back in core. If all conditions as specified by the legality flags are met, control is passed to the appropriate program.

Table VIII. File Protection Scheme.

Protection Level	Access Code	Access Privileges
Greatest Protection 	7	No access privileges
	6	EXECUTE ONLY
	5	READ, EXECUTE
	4	APPEND, READ, EXECUTE
	3	UPDATE, APPEND, READ, EXECUTE
	2	WRITE, UPDATE, APPEND, READ, EXECUTE
	1	RENAME, WRITE, UPDATE, APPEND, READ, EXECUTE
	0	CHANGE PROTECTION, RENAME, WRITE, UPDATE, APPEND, READ, EXECUTE
Least Protection		

File Handler

To reference a file, the user does not need to know where the file is physically located. A named file is uniquely identified in the system by a file name and extension, an ordered list of directory names (UFDs and SFDs) which identify the owner of the file, and a file structure name which identifies the group of disk units containing the file.

Usually a complete disk system is composed of many disk units of the same and/or different types. Therefore, the disk system consists of one or more file structures—a logical arrangement of files on one or more disk units of the same type. This method of file storage allows the user to designate which disk unit of the file structure he wishes to use when storing files. Each file structure is logically complete and is the smallest section of file memory that can be removed from the system without disturbing other units in other file structures. All pointers to areas in a file structure are by way of logical block numbers rather than physical disk addresses; there are no pointers to areas in other file structures, thereby allowing the file structure to be removed.

File Structures

A file structure contains two types of files; the data files that physically contain the stored data or programs, and the directory files that contain pointers to the data files. Included in these directory files are master file directories, user file directories, and sub-file directories. Each file structure has one master file directory (MFD). This directory file is the master list of all the users of the file structure. The entries contained in the MFD are the names of all the user file directories on the file structure. Each user with access to the file structure has a user file directory (UFD) that contains the names of all his files on that file structure; therefore, there are many UFDs on each file structure. As an entry in the user file directory, the user can include another type of directory file, a sub-file directory (SFD). The sub-file directory is similar to the other types of directory files in that it contains as entries the names of all files within that sub-directory. This third level of directory allows groups of files belonging to the same user to be separate from each other. This is useful when organizing a large number of files according to function. In addition, sub-file directories allow non-conflicting, simultaneous runs of the same program using the same file names.

As long as the files are in different sub-file directories, they are unique. Sub-file directories exist as files pointed to by the user file directory, and can be nested to the depth specified by the installation at system generation time.

File Protection

All disk files are composed of two parts: data and information used to retrieve data. The retrieval part of the file contains the pointers to the entire file, and is stored in two distinct locations on the device and accessed separately from the data. System reliability is increased with this method because the probability of destroying the retrieval information is reduced; system performance is improved because the number of positionings needed for random-access methods is reduced. The storing of retrieval information is the same for both sequential and random-access files. Thus a file can be created sequentially and later read randomly, or vice versa, without any data conversion.

One section of the retrieval information is used to specify the protection associated with the file. This protection is necessary because disk storage is shared among all users, each of whom may desire to share files with, or prevent files from being written, read, or deleted by other users. As discussed above, these protection codes are assigned by the user when the file is created and designate the users who have privileges to access the file.

Disk Quotas

Disk quotas are associated with each user (each project-programmer number) on each file structure in order to limit the amount of information that can be stored in the UFD of a particular file structure. When the user gains access to the computing system, he automatically begins using his logged-in quota. This quota is not a guaranteed amount of space, and the user must compete with other users for it. When the user leaves the computing system, he must be within his logged-out quota. This quota is the amount of disk storage space that the user is allowed to maintain when he is not using the system and is enforced by the system program that is used in logging off the system. Quotas are determined by the individual installation and are, therefore, used to ration disk resources in a predetermined manner.

To the user, a file structure is like a device; i.e., a file structure name or a set of file structure names can be used as the device name in command strings or UUO calls to the operating system. Although file structures or the units composing the file structures can be specified by their actual names, most users specify a general, or generic, name (DSK) which will cause the operating system to select the appropriate file structure. The appropriate file structure is determined by a job search list. Each job has its own job search list with the file structure names in the order in which they are to be accessed when the generic name is specified as the device. This search list is established by LOGIN and thus each user has a UFD for his project-programmer number in each file structure in which LOGIN allows him to have files.

File Operations

File writing on the disk can be defined by one of four methods: creating, superseding, updating and simultaneous updating. The user is creating a file if no other file of the same name exists in the user's directory on the indicated file structure. If another file with the same name already exists in the directory, the user is superseding, or replacing, the old file with the new file. Other users sharing the old file at the time it is being superseded continue using the old file and are not affected until they finish using the file and then try to access it again. At that time, they read the new file. When a user updates a file, he modifies selection parts of the file without creating an entirely new version. This method eliminates the need to recopy a file when making only a small number of changes. If other users try to access a file while it is being updated, they receive an error indication issued by the system.

Many users can update the same file through the use of the simultaneous update feature. This allows users to work with the same data base simultaneously. An ENQ/DEQ facility is also offered to allow record level lock out and thus eliminate race conditions where multiple users are attempting to read and/or write the same record.

Disk Storage Management

File storage is dynamically allocated by the file handler during program operations, so the user does not need to give initial estimates of file length or the number of files. Files can be any length, and each user may have as many files as he wishes, as long as disk space is available and the user has not exceeded his logged-in quota. This feature is extremely useful during program development or debugging when the final size of the file is still unknown. However, for efficient random access, a user can reserve a contiguous area on the disk if he desires. When he has completed processing, he can keep his preallocated file space for future use or return it so that other users can have access to it.

Input / Output

Peripheral Device Assignment

With the command language, the user can also request assignment of any peripheral device (magnetic tape, DECtape, and private disk pack) for exclusive use. When the request for assignment is received, the operating system verifies that the device is available to this user, and the user is granted its private use until he relinquishes it. In this way, the user can also have complete control of devices such as card readers and punches, paper tape readers and punches, and line printers.

Spooling

When private assignment of a slow-speed device (card punch, line printer, paper tape punch, and plotter) is not required, the user can employ the spooling feature of the operating system. Spooling is a method by which output to a slow-speed device is placed on a high-speed disk or drum. This technique prevents the user from consuming unnecessary system resources while waiting for either a device to become available or output to be completed. In addition, the device is managed to a better degree because the users cannot tie it up indefinitely, and the demand fluctuations experienced by these devices are equalized.

Memory Management

The DECsystem-10 is a multiprogramming system; i.e., it allows multiple independent user programs to reside simultaneously in memory and to run concurrently. This technique of sharing memory and processor time enhances the efficient operation of the system by switching the processor from a program that is temporarily stopped because of I/O transmission to a program that is executable. When core and the processor are shared in this manner, each user's program has a memory area distinct from the area of other users. Any attempt to read or change information outside of the area a user can access immediately stops the program and notifies the user.

Because available memory can contain only a limited number of programs at any one time, the computing system employs a secondary memory, usually disk or drum, to increase the number of users serviced. User programs exist on the secondary memory and move into memory for execution. Programs in memory exchange places with the programs being transferred from secondary memory for maximum use of available main memory. Because the transferring, or swapping takes place directly between main memory and the secondary memory, the central processor can be operating on a user program in one part of memory while swapping is taking place in another. This independent, overlapped operation greatly improves system utilization by increasing the number of users that can be accommodated at the same time.

To further increase the utilization of memory, the operating system allows users to share the same copy of a program or data segment. This prevents the excessive memory usage that results when a program is duplicated for several users. A program that can be shared is called a reentrant program and is divided into two parts or segments. One segment contains the code that is not modified during execution (e.g., compilers and assemblers) and can be used by any number of users. The other segment contains non-reentrant code and data. The operating system provides protection for shared segments to guarantee that they are not accidentally modified.

Virtual Memory

The virtual memory option permits a user program to execute with an address space greater than the physical memory actually allocated to that program during execution. User jobs are swapped as described above. However, the entire program may not necessarily be in core during execution. Programs are divided into pages each of which is 512-words long. Some of these pages may remain on secondary storage while the program executes. When a virtual memory job attempts to access a page that is not in core, a page fault handler decides which page or pages to remove from core and which to bring in from secondary storage.

Unlike the virtual memory implementation on other systems, this DECsystem-10 feature is an option. Each site may determine its own need for virtual memory and install it at their convenience. The system administrator may grant the privilege for using virtual memory only to those users who truly need its capabilities. Those users who are granted the privilege of using virtual memory may elect to invoke the feature for only those programs that could not execute without the virtual memory capability.

The virtual memory users may elect to use the system page fault handler or they may use a handler that is more tailored to the particular application or program behavior. Finally, it is important to point out that only those users actually using the virtual memory feature are affected by any additional overhead associated with a demand paging system. Non-virtual users execute as they would in a non-virtual system with no discernible difference in performance.

Multi-Processor Systems

DECsystem-10 Dual Processor systems are composed of two CPU's, designated the primary processor (master) and the secondary processor (slave). The primary processor is connected to all of the memory in the system and has all of the system's peripheral I/O equipment connected to its I/O bus. The secondary processor also has access to all of memory, however, there are normally no I/O devices on this processor's I/O bus. The primary processor performs exactly the same operations as the processor in a single processor system. This includes all I/O operations, swapping, core allocation, resource allocation and command decoding. The secondary processor also performs scheduling and execution of user jobs according to the same algorithm used in a single processor system.

The secondary processor executes user jobs and scans the same job queues as the primary processor. However, since the slave cannot do any standard I/O, it looks for any compute bound jobs which are in core and runnable. A software interlock has been added to the scheduler to prevent the possibility of both processors trying to execute the same job at the same time. Whenever a job being executed by the secondary processor requests an I/O operation to be performed the job is stopped and marked for execution on the master only. Thus both processors run completely asynchronously, both executing the same scheduler, doing the same job accounting and using the same job queues.

The existence of dual processors gives DECsystem-10 users a large scale computing capability, especially in the areas of highly compute bound jobs and non-interactive batch jobs. The existence of a slave more or less dedicated to user computation allows these jobs to be carried on in the systems with little interference with time sharing users. The performance goal is to provide a system in which each customer can improve the service offered to his users over a single processor compute bound system. Either the customer can add more users with the same response time or he can keep the same number of users and reduce turn around time for compute bound jobs.

Inter-job Communication

Shared Data Areas

The DECsystem-10 operating system enables a user to communicate with other jobs through the use of sharable data areas. This also enables a data analysis program, for example to read or write an area in the real-time job's core space. Since the real-time job associated with the data acquisition would be locked in core, the data analysis program residing on secondary memory would become core resident only when the real-time job had filled a core buffer with data. Operating system calls can be used to allow the data analysis program to remain dormant in secondary memory until a specified event occurs in the real-time job: e.g., a buffer has been filled with data for the data analysis.

Inter-Process Communication Facility

The Inter-Process Communication Facility (IPCF) provides the capability for independent jobs to communicate with one another. For example, if several programs are involved in processing or maintaining a data base, it is possible that one program might want to inform the others of any modifications it made to the data. A job using IPCF cannot make any changes to another job so protection is in no way sacrificed when using the IPCF feature.

In order to use the IPCF, each participating job that wishes to receive communication from other jobs must request a unique process identifier (PID) from the system. The transmitting job then may send a "packet" of information to another job. (In addition to the information, the system automatically provides a "return address" so that the receiving program can respond to the sender.) The monitor maintains a linear queue (the "mailbox") for each job using IPCF. The packet (or packets) will be kept in the mailbox until the receiving job retrieves it. This queue is not created until a job sends an IPCF packet and it does not occupy any space until such time. The maximum number of packets allowed in a queue at any one time is determined by a "receive" quota that may be set at each installation for each user. (If no quota is set by the installation, the standard default is five.)

On systems with the virtual memory option, the packet could be an entire page. In this case, the monitor takes advantage of the page mapping hardware of the K110 and KL10 to transmit the page without actually copying it.

Communications Software

DECsystem-10 users have a wide selection of communications products to enhance or facilitate their computing needs. Within the multi-task environment of TOPS-10, functionality exists for:

- Asynchronous communication—the typical vehicle for interactive time-sharing or transaction-oriented terminals
- Synchronous communications—for connection of remote batch, remote terminal concentration and computer-to-computer links.

The important feature of DECsystem-10 communications software is its implementation as an integral part of the TOPS-10 Operating System. The monitor, not the user, handles the communications housekeeping, and all communications products are fully supported within the TOPS-10 environment. Appropriate synchronous line protocols—DIGITAL's own DDCMP or BISYNC—are supported.

Users may configure networks with simple or complex topologies utilizing the DN87 family of Universal Front End equipment. The DAS-80 and DAS-90-series remote stations/concentrators can then be used to expand the network topologies. In addition, DECNET-10 permits the DECsystem-10 to utilize task-to-task communication with other DECsystem-10's or with any other DEC-supplied computer system which supports DECNET. The DAS-61 and DAS-62 can be used to do IBM 2780, 3780 or HASP multi-leaving emulation and termination.

Additional details concerning the DECsystem-10 communications capability and products are included in the Data Communications Section.

Multimode Computing

Timesharing

Timesharing takes maximum advantage of the capabilities of the computing system by allowing many independent users to share the facilities of the DECsystem-10 simultaneously. Because of the interactive, conversational, rapid-response nature of TOPS-10 time-sharing, a wide range of tasks—from solving simple mathematical problems to implementing complete and complex information gathering and processing networks—can be performed by many users concurrently. The number of users on the system at any one time depends on the system configuration and the job mix on the system. DECsystem-10 timesharing is designed to support in excess of 200 active users. Interactive terminals can include CRTs, hard copy terminals and other devices which operate at speeds of 110 to 9600 baud. Terminal users can be located at the computer center or at remote locations connected to the computer center by communication lines.

The DECsystem-10 is designed for the concurrent operation of timesharing, multistream batch, real-time, and remote communications in either single- or dual-processor systems. In providing these multifunction capabilities, the DECsystem-10 services interactive users, operates local and remote batch stations, and performs data acquisition and control functions for on-line laboratories and other real-time projects. By dynamically adjusting system operation, the DECsystem-10 provides many features for each class of user and is therefore able to meet a wide variety of computational requirements.

Timesharing on the DECsystem-10 is general purpose; i.e., the system is designed in such a way that the command language, input/output processing, file structures, and job scheduling are independent of the programming language being used. In addition, standard software interfaces make it easy for the user to develop his own special language or systems. This general purpose approach is demonstrated by the many special-purpose programming languages implemented by DECsystem-10 users.

GALAXY-10 Batch

GALAXY-10 batch software enables the DECsystem-10 to execute in excess of 30 batch jobs concurrently with timesharing jobs. Just as the timesharing user communicates with the system by way of his terminal, the batch user normally communicates by way of the card reader. (However, he can also enter his job from an interactive terminal.) Unlike the timesharing user, the batch user can punch his job on cards, insert a few appropriate control cards, and leave the job for an operator to run. In addition, the user can debug the program in the time sharing environment and then run it in batch mode without any additional coding.

The GALAXY-10 system consists of a series of programs: QUASAR, the system queue manager and scheduler, SPRINT, the input spooler, BATCON, the batch controller, and the output spoolers, LPTSPL and SPROUT. The input spooler is responsible for reading the input from the input device and for entering the job into the batch controller's input queue. The input spooler sends a message to QUASAR to do the actual entering into the batch input queue. Although the input spooler is oriented toward card reader input, disk and magnetic tape also can be handled. The input information is then separated according to the control commands in the input deck and placed into disk files 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 its processing. The log file is part of the standard output that the user receives when his job terminates.

After the input spooler reads the end-of-file and closes the disk files, it makes an entry in the batch controller's input queue. The batch controller processes batch jobs by reading the entries in its queue. The control file created by the input spooler is read by the batch controller, and data and user program commands are passed directly to the user's job. Operating system commands are detected by the batch controller and passed to the operating system for action. Most operating system and user program commands available to the timesharing user are also available to the batch user. Therefore, only one control language need be learned for both timesharing and batch. During the processing of the job and the control file, the batch controller adds information to the log file for later analysis by the user.

QUASAR 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 under the batch controller according to external priorities, processing time limits, and core requirements which are dynamically computed according to parameters specified by the user for his job, such as start and deadline time limits for program execution. QUASAR makes an entry for the job in the batch input queue based upon the various priorities. After the job is completed an entry is made in the output queues for the job's spooled output and the job's LOG file.

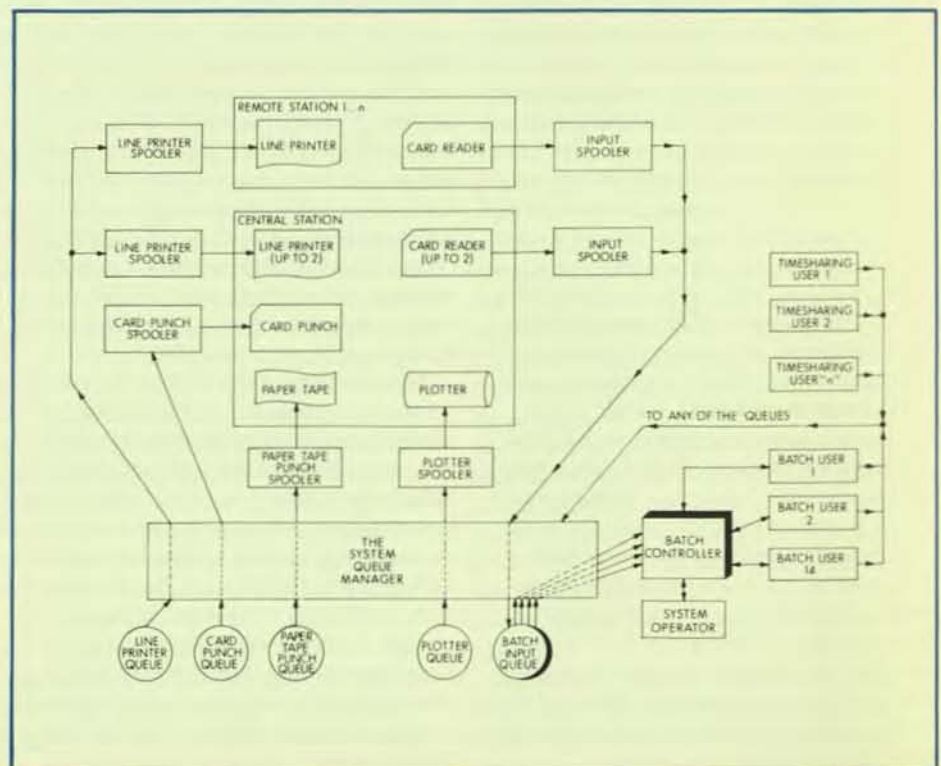
The output spooling programs improve system throughput by allowing the output from a job to be written temporarily on the disk for later transfer instead of being written immediately on a particular output device. The log file and all job output are placed into one or more output queues to await processing. When the specified device is available, the output is then processed by the appropriate spooling program. These spooling programs may be utilized by all users of the computing system.

Use of System Features

The GALAXY-10 batch software employs many of the computing system's features in order to operate with maximum efficiency. Because core memory is not partitioned between batch and timesharing jobs, batch jobs can occupy any available area of core. Fast throughput for high-priority batch jobs is accomplished with the same swapping technique used for rapid response to interactive users. When available core is not large enough for a high-priority batch job, the operating system transfers programs of lower priority to secondary memory in order to provide space for the job. This transfer is done at the same time that the processor is operating on another job. Thus, processing can be overlapped with swapping (and other I/O) to utilize time that would otherwise be wasted. Batch jobs can also share programs with timesharing and other batch jobs. Only one copy of a sharable program need be in core to service any number of batch and timesharing jobs at the same time.

Flexibility

GALAXY-10 batch allows the user great flexibility. The input spooler normally reads from the card reader, but can read from magnetic tape or disk in order to create a control file on disk and to enter the job into the batch controller's input queue. However, a job can be entered from an interactive terminal, in which case the user bypasses the input spooler and creates a control file on disk for the batch controller. The control file contains the operating system commands and user program commands necessary to run the job. The user then enters the job into the batch controller's input queue by way of an operating system command string. In the command string, the user can include switches to define the operation and set the priorities and limits on core memory and processor time.



Job Dependency

Although jobs are entered sequentially into the batch system, they are not necessarily run in the order that they are read because of priorities either set by the user in an input spooler control command or computed by the queue manager when determining the scheduling of jobs. Occasionally, the user may wish to submit jobs that must be executed in a particular order; in other words, the execution of one job is dependent on another. To ensure that jobs are executed in the proper order, the user must specify an initial dependency count in a control command of the dependent job. This dependency count is then part of the input queue entry. A control command in the job on which the dependent job depends decrements the count. When the count becomes zero, the dependent job is executed.

Error Recovery

The user can control system response to error conditions by including commands to the batch controller which will aid in error recovery. These commands are copied into the control file by the input spooler. With error recovery commands, the user specifies the action to be taken when his program contains a fatal error, as for example, to skip to the next program or to transfer to a special user-written error handling routine. If an error occurs and the user did not include error recovery conditions in his job, the batch controller initiates a standard dump of the user's core area and terminates the job. This core dump provides the user with the means to debug his program.

Although the batch system allows a large number of parameters to be specified, it is capable of operating with very few user-specified values. If a parameter is missing, the batch system supplies a reasonable default value. These defaults can be modified by the individual installations.

Operator Intervention

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

Real-time Computing

For a system to be satisfactory for real-time operations, two important requirements must be met. The more important requirement is fast response time. Because real-time devices may not be able to store their information until the computing system is ready to accept it, the system would be useless for real-time if the response requirements of a real-time project could not be satisfied. The operating system allocates system resources dynamically in order to satisfy the response and computational requirements of real-time jobs.

The second requirement is protection. Each user of the computing system must be protected from other users, just as the system itself is protected from all user program errors. In addition, since real-time systems have special real-time devices associated with jobs, the computing system must be protected from hardware faults that could cause system breakdown. And, because protection is part of the function of the operating system, the real-time software employs this feature to protect users as well as itself against hardware and software failures. Inherent in the operating system is the capability of real-time, and it is by way of calls to the operating system that the user obtains real-time services. The services obtained by calls within the user's program include:

- Locking a job in core.
- Connecting a real-time device to the priority interrupt system.
- Placing a job in a high-priority run queue.
- Initiating the execution of FORTRAN or machine language code on receipt of an interrupt.
- Disconnecting a real-time device from the priority interrupt system.

Locking Jobs

Memory space is occupied by the resident operating system and by a mix of real-time and non-real-time jobs. The only fixed partition is between the resident operating system and the remainder of memory so as not to lose information when its associated real-time device interrupts (since there may not be sufficient time to swap-in the job) the job can request that it be locked into core. This means that the job is not to be swapped to secondary memory and guarantees that the job is readily available when needed. Because memory is not divided into fixed partitions, it can be utilized to a better degree by dynamically allocating more space to real-time jobs when real-time demands are high. As real-time demands lessen, more memory can be made available to regular timesharing and batch usage.

Real-time Devices

The real-time user can connect real-time devices to the priority interrupt system, respond to these devices at interrupt level, remove the devices from the interrupt system, and/or change the priority interrupt level on which these devices are assigned. There is no requirement that these devices be connected at system generation time. The user specifies both the names of the devices generating the interrupts and the priority levels on which the devices function. The operating system then links the devices to the operating system.

The user can control the real-time device in one of two ways: single mode or block mode. In single mode, the user's interrupt program is run every time the real-time device interrupts. In block mode, the user's interrupt program is run after an entire block of data has been read from the real-time device. When the interrupt occurs from the device in single mode or at the end of a block of data in block mode, the operating system saves the current state of the machine and jumps to the user's interrupt routine. The user services his device and then returns control to the operating system to restore the previous state of the machine and to dismiss the interrupt. Any number of real-time devices may be placed on any available priority interrupt channel for the program to read. When the specified event occurs, the dormant program is then activated to process the data. The core space for the real-time job's buffer area or the space for the dormant job does not need to be reserved at system generation time. The hardware working in conjunction with the operating system's core management facilities provides optimum core usage.

High-Priority Run Queues

The real-time user can receive faster response by placing jobs in high-priority run queues. These queues are examined before all other run queues in the computing system, and any runnable job in a high-priority queue is executed before jobs in other queues. In addition, jobs in high-priority queues are not swapped to secondary memory until all other queues have been scanned. When jobs in a high-priority queue are to be swapped, the lowest priority job is swapped first and the highest priority job last. The highest priority job swapped to secondary memory is the first job to be brought into core for immediate execution. Therefore, in addition to being scanned before all other queues for job execution, the high-priority queues are examined after all other queues for swapping to secondary memory and before all other queues for swapping from secondary memory.

Diagnostic Software

"High Availability"

TOPS-10 provides a number of features which allow the system to continue operating although some devices may be inoperable.

- 1) When errors occur on disks, tapes, memory, or the slave CPU, TOPS-10 will record the error data and notify the operator. If the error can be fixed (e.g., put a disk back on-line), the user may continue running without further delay. Alternatively, the operator may instruct the system to remove (detach) the device, in which case the jobs in error will be notified and the system will continue operation without the specified device. If the error is in a bank of memory, TOPS-10 will even attempt to move its own code (if any) out of the failing memory. In the case of disks, the system will attempt to migrate any swapping space to another unit to prevent loss of jobs.
- 2) RP04, and RP06 type disks may be dual ported so that in case of controller failure, the packs may be accessed via an alternate path. In case of a unit failure, the pack may be moved to a working unit.
- 3) With use of appropriate I/O bus, memory bus, and device control switches, DECsystem-10's may be configured to provide redundancy of hardware. Such systems may allow any piece of hardware to be taken off-line and repaired while the system continues operation (in some cases, it may be necessary to reload the system after a significant reconfiguration).

Maintenance Features

The Diagnostic Strategy has produced a high quality of maintenance aids which enforce the increased performance and availability of a DECsystem-10. The design of the KL10 processor uses integrated diagnostic logic to enable maximum visibility and reduce "hard-core" requirements. Special software techniques are employed to maximize test coverage per diagnostic load, thereby decreasing the mean time to diagnose.

Board Level Diagnosis

The Writable Control Store feature of the KL10 has enabled the "Micro Diagnostic" alternate path methodology of testing. This technology has enabled generation of diagnostics which narrow malfunctions to the board level with a degree of confidence which exceeds industry standards.

Maintainability was attacked from a total system's concept, for not only are the CPU diagnostics capable of supporting the "Module Swap" philosophy of repair, the new generation of peripheral diagnostics gives the maintainer the same qualities. The programs utilize special diagnostic wrap around logic to isolate suspected malfunctions to subsystem level. Then, through comprehensive logic analysis, not just functionality, the repair person is directed to the failing module.

On-line Diagnosis

To further increase system availability, special features have been incorporated into the monitor to enable on-line preventive maintenance.

Emphasis has also been placed on testing the thoroughness of detectability and the correctness of isolation. Semiautomatic physical fault insertion is qualifying and providing maturity enhancements for this product.

KLINIK Remote Diagnosis

One final feature of the Maintenance and Diagnostic Philosophy is the support of remote diagnosis (KLINIK). This can provide distant maintenance personnel with hardware performance statistics to improve the efficiency of service on KL10 Based Systems.

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.

SERVICES	85
Hardware Maintenance	87
People	87
Parts	87
Support	87
Service Agreements	88
Education	88
Logistics	89
Technical Documentation	89
Software Services	90
Advanced Systems Group	91

Hardware Maintenance

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; and a wide range 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 intensive 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 our own 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.

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 90 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.

Logistics

Good service demands outstanding logistics. Spares availability minimizes 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-10 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.

Technical Documentation

DIGITAL is a major publisher of manuals, handbooks, and engineering literature. These documents support more than 65,000 installed DIGITAL computers. They meet user requirements for installation, maintenance, education and operation. They provide an authoritative guide to DIGITAL systems.

One example is DEC-O-LOG. Maintaining current systems and documents is a critical task. DEC-O-LOG satisfies this requirement by listing and explaining engineering improvements as they evolve. This information is available to keep you abreast of the latest technological developments.

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 65,000 computer installations. DIGITAL's software varies in complexity from real-time executives for minicomputers to the TOPS-10 Monitor of the DECsystem-10. 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-10 and keep pace with advancements in software.

You purchase only what you need. Software components, including manuals and updates, can be purchased as part of a Software Maintenance Service or ordered separately from the Software Distribution Center. Software maintenance plans for the DECsystem-10 include:

1. Software Notebook Updates—Each customer receives two sets of comprehensive software notebooks which can be kept current via periodic update packages. You are assured of always having the most current documentation published.
2. Program Updates—Subscribers receive the latest software versions along with all available "bug-fixes" on magtape.
3. Software DISPATCH—This service provides all available code corrections as well as a copy of all customer-generated Software Performance Reports published as an early warning service. This is a comprehensive publication aimed at the customer in a dynamic, growing software environment.
4. Expedited Software Performance Report Service—This service provides a very fast priority response system with personalized replies returned directly to the originator, thus bypassing normal lags associated with publication and program update cycles. This insures that any problems you encounter will be corrected.
5. Consulting Service—These are available on a short-term, per-call basis or for a longer term scheduled or resident period. Consultants can install software updates, train new employees, and recommend procedures that result in top performance by your software system.

Advanced Systems Group

The Advanced Systems Group, an integral part of the DECsystem-10 product group, meets special customer's needs by augmenting the standard products and services of the DECsystem-10. The Advanced Systems Group extends the range of the DECsystem-10 product and services by providing:

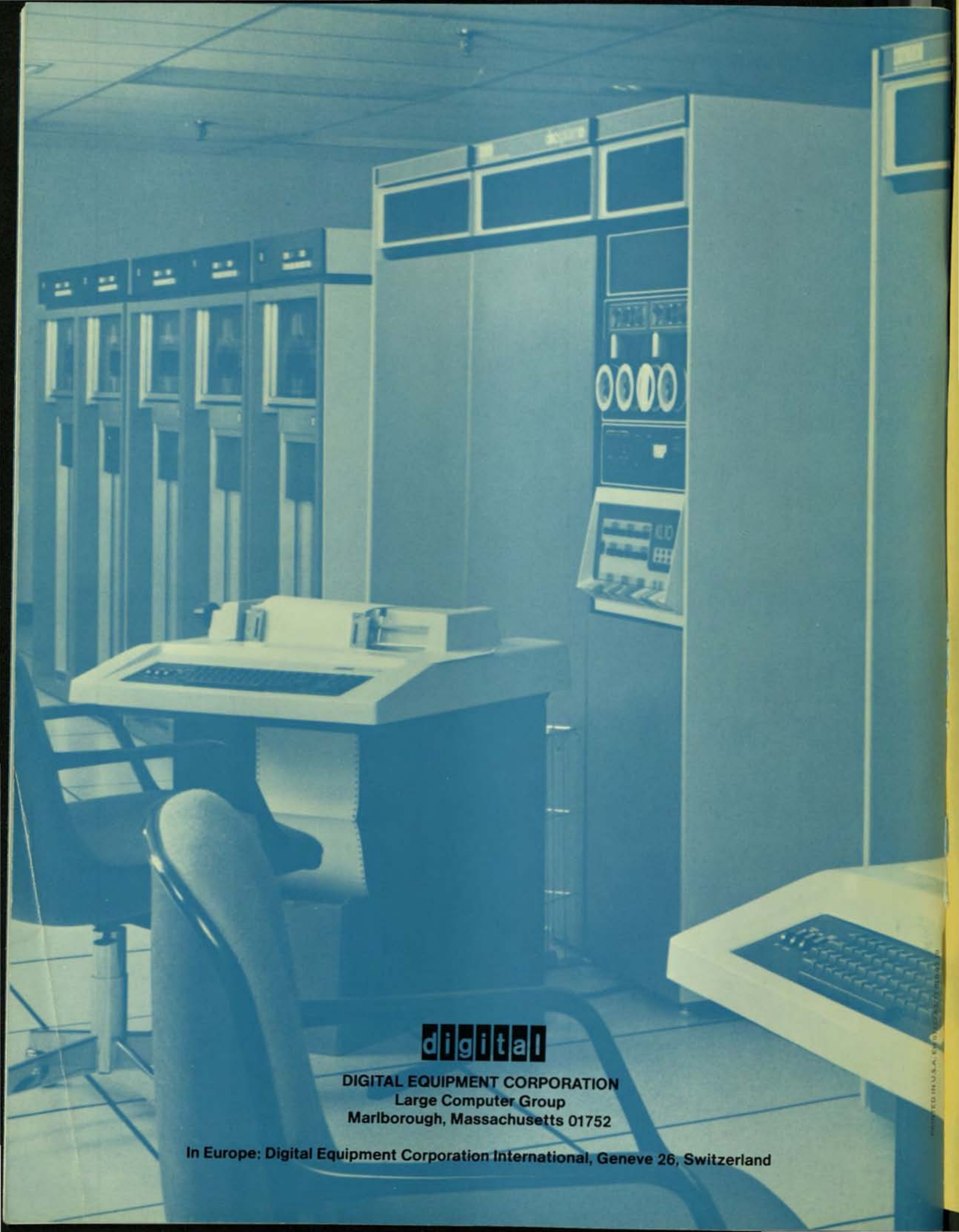
1. Hardware/Software Design and Development—Engineering for special products and modifications to standard products, with emphasis in the areas of communications, network systems, high availability and redundant systems, and special configurations.

2. 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-10 acceptance test procedures.
3. 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-10 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.



digital

DIGITAL EQUIPMENT CORPORATION
Large Computer Group
Marlborough, Massachusetts 01752

In Europe: Digital Equipment Corporation International, Geneva 26, Switzerland

ToLitt

TECHNICAL SUMMARY

digital

DECSYSTEM
20

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Digital Equipment Corporation makes no representation that the interconnection of its products in the manner described herein will not infringe on existing or future patent rights, nor do the descriptions contained herein imply the granting of license to make, use, or sell equipment constructed in accordance with its description.

The software described in this document is furnished under license and may be used or copied only in accordance with the terms of such license.

Unless otherwise noted, $K = 1,024$; $M = K^2$.

DEC, DECUS, the Digital logo, PDP, VAX, DECnet, DECwriter, DECsystem, DECSYSTEM, DIBOL, RSTS, RSX, UNIBUS, MASSBUS, and VT are trademarks of Digital Equipment Corporation.

Copyright© 1981 Digital Equipment Corporation. All rights reserved.

Contents

1 PREFACE

2 OVERVIEW

COMPONENTS	2-1
Processors	2-1
TOPS-20 Operating System	2-1
RELIABILITY	2-3
Software Design	2-3
System Availability	2-3
System Recovery	2-3
Error Reporting	2-3

3 THE USERS

THE APPLICATION PROGRAMMER	3-1
The Programming Languages	3-1
Application Tools	3-1
THE SYSTEM PROGRAMMER	3-1
THE SYSTEM MANAGER	3-2
User Authorization	3-2
Privileges	3-2
Allocating Disk Storage Quotas	3-2
Controlling Resources	3-2
Scheduler Controls	3-2
Resource Accounting Statistics	3-3
Performance Analysis Statistics	3-3
THE SYSTEM OPERATOR	3-3
Operator Interface	3-3
Controlling Batch Streams	3-3
System Recovery	3-3
System Backup	3-3
USER UTILITIES	3-3

4 THE OPERATING SYSTEM

THE COMMAND LANGUAGE PROCESSOR	4-1
THE BATCH SYSTEM	4-1
JOBS AND PROCESSES	4-2
PROCESS COMMUNICATION	4-2
Direct Process Control	4-2
Interprocess Communication Facility	4-2
Software Interrupt System	4-2
Enqueue/Dequeue (ENQ/DEQ)	4-3
Memory Sharing	4-3
THE TOPS-20 MONITOR	4-3
VIRTUAL MEMORY	4-4
VIRTUAL MEMORY OPERATIONS	4-4
THE FILE SYSTEM	4-5
Files	4-5
File Protection	4-5
File Directories	4-6
Groups	4-6
File Usage	4-6
Archiving Files	4-6
Tape Labelling	4-6
CONSOLE FRONT-END PROCESSOR AND SOFTWARE	4-6
Console Functions	4-6
Command Terminal Functions	4-6
Peripheral Interface	4-6
Diagnostic/Maintenance Functions	4-6

5 KL10-E PROCESSOR

EXECUTION BOX (EBOX)	5-1
Instruction Set	5-2
Meters	5-4
Priority Interrupt System	5-4

MEMORY SUBSYSTEM	5-5
Internal Memory	5-5
MOS Memory	5-5
Cache Memory	5-5
Organization	5-6
System Control of Cache	5-7
Memory Mapping on the KL10-E	5-7
FRONT-END SUBSYSTEM	5-7
INPUT/OUTPUT SUBSYSTEM	5-8
Multiplexed I/O Bus	5-8
MASSBUS	5-9
UNIBUS	5-9

6 KS10 PROCESSOR

SYSTEM ARCHITECTURE	6-1
KS10 Technology	6-1
The KS10 Central Processing Unit	6-1
Cache Memory	6-2
General Registers	6-2
Microstore	6-2
Instruction Set	6-2
Processor Modes	6-3
MEMORY	6-3
Memory Address Mapping	6-3
MOS Memory	6-4
MS10 memory	6-4
CONSOLE SUBSYSTEM	6-4

7 THE PERIPHERALS

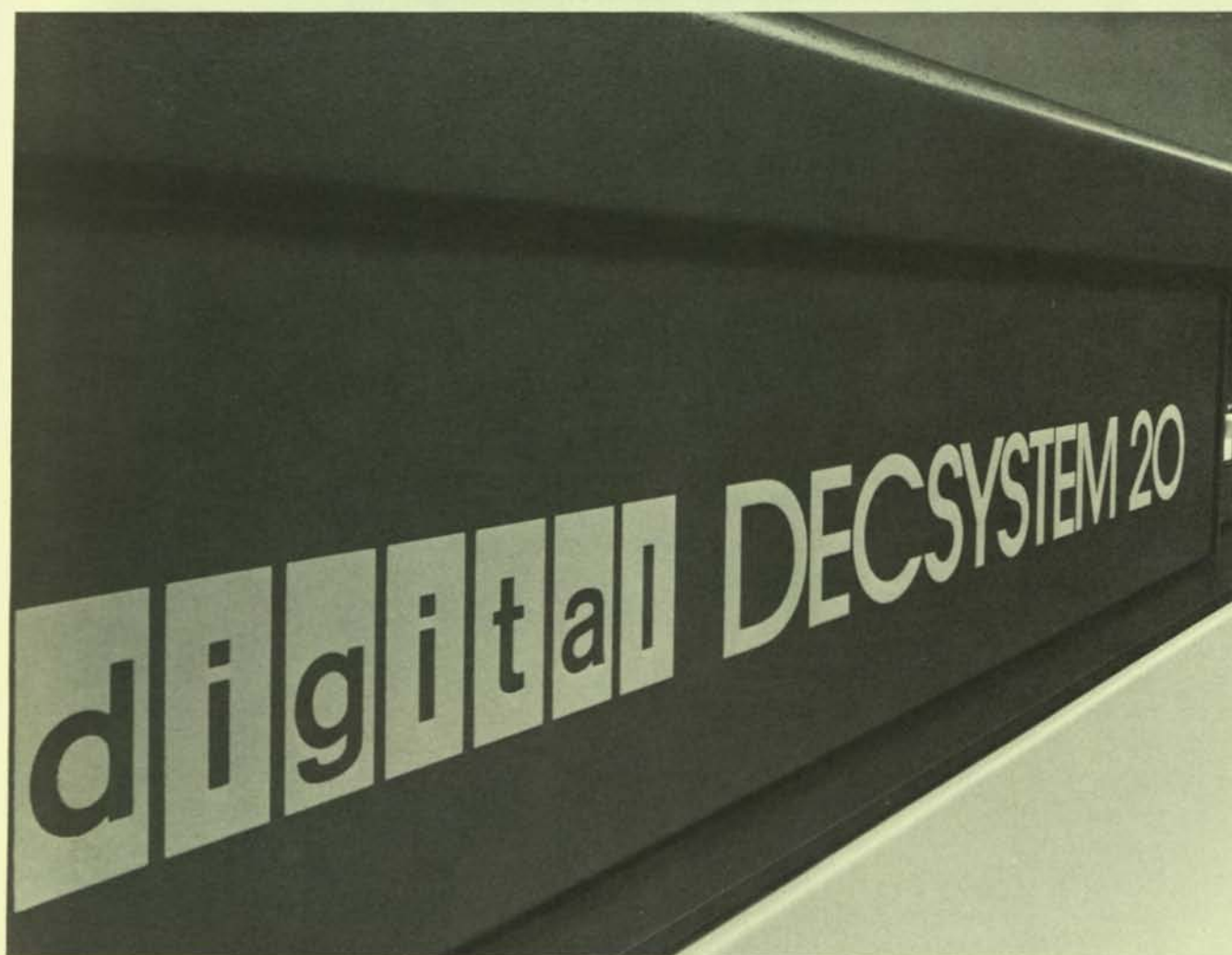
MASS STORAGE PERIPHERALS	7-1
Disks	7-1
Tape Devices	7-1
UNIT RECORD PERIPHERALS	7-2
LP20-A and -B Lineprinters	7-2
LP20-C and -D Lineprinters	7-3
LP200 Lineprinters	7-3
Cardreaders	7-4
PC20 Paper Tape Reader/Punch	7-4
TERMINALS AND INTERFACES	7-4
LA120 Hardcopy Terminal	7-4
LA38 Hardcopy Terminal	7-5
VT100 Video Terminal	7-5
DZ11 Terminal Line Interface	7-6
DH11 Programmable Asynchronous Serial Line Multiplexer	7-6
DL11 Serial Line Asynchronous Interfaces	7-6
DUP11 Single Synchronous Line Interface	7-6
KMC11-A Auxiliary Processor	7-6
DN20 COMMUNICATIONS FRONT END	7-6

8 THE LANGUAGES

TOPS-20 ASSEMBLER	8-1
DEBUGGING TOOLS	8-1
FORTRAN	8-1
Language Extensions	8-2
Optimization	8-2
Debugging Tools	8-2
COBOL	8-2
Data Types	8-3
String Manipulation	8-3
Interactive COBOL Execution	8-3
File Organization	8-3
Library Facility	8-3
ENTER Facility	8-3

Online Debugger	8-3	Data Manipulation Process	9-1
Source Program Input	8-3	DBMS Modules	9-1
RERUN	8-4	DBMS Utilities	9-2
COBOL-68 and COBOL-74	8-4	IQL	9-2
ALGOL-20	8-4	Interactive Mode	9-2
Block Structure	8-4	Deferred Mode	9-2
Procedures	8-4	IQL Statements	9-3
Compiler and System Features	8-5	DBMS Files	9-3
OWN Variables	8-5	Report Formatting	9-3
Switches	8-5	SORT/MERGE	9-4
String Constants	8-5	TRAFFIC-20	9-4
Object-Time System	8-5	COGO-20	9-4
BASIC-PLUS-2	8-5	PCS-20	9-4
File Capability	8-6		
Virtual Memory Arrays	8-6	10 COMMUNICATIONS	
Record Input/Output	8-6	DECNET-20	10-1
Data Formats and Operations	8-6	DIGITAL Network Architecture	10-1
Matrix Manipulation	8-7	DECnet Functions	10-2
Conditional Statements and Program Segmentation	8-7	DECnet-20 Capabilities	10-2
Debugging Tools	8-7	REMOTE JOB ENTRY STATIONS	10-3
APL	8-7	INTERNET PROTOCOL EMULATORS	10-3
Data Structures	8-8	DX/TOPS-20 DOCUMENT TRANSMISSION	10-5
Interacting with APL	8-8		
System Commands, Functions, and Variables	8-8	11 SUPPORT SERVICES	
Statements	8-8	INSTALLATION	11-1
APL Statement Execution	8-8	SOFTWARE SERVICES	11-1
Debugging Tools	8-9	Software Warranty	11-1
Workspaces	8-9	Software Product Services	11-1
File Organization	8-9	Professional Services	11-1
Error Analysis and Recovery	8-9	EDUCATIONAL SERVICES	11-1
Conversion Package	8-9	Course Options	11-1
BLISS-36	8-9	TOPS-20 Courses	11-2
Compiling	8-9	HARDWARE SERVICES	11-3
Debugging	8-9	COMPUTER SPECIAL SYSTEMS	11-3
File Organization	8-9	CUSTOMER FINANCING	11-4
Compatibility with Other Languages	8-10	ACCESSORIES AND SUPPLIES GROUP	11-4
CPL	8-10	COMPUTER SUPPLIES	11-4
Immediate Mode	8-10	CUSTOMER SPARES	11-4
Program Creation	8-10	DECUS	11-5
Debugging	8-10		
		12 GLOSSARY	
9 DATA MANAGEMENT AND APPLICATION PRODUCTS		13 INDEX	
DBMS	9-1		
CODASYL Compliance	9-1		
Data Description Process	9-1		

1 Preface



This technical summary is a detailed introduction to all aspects of the DECSYSTEM-20 — from the processors and peripheral devices to the TOPS-20 software and DIGITAL's support services. The technical summary is primarily intended for system programmers and computer system specialists already familiar with computer hardware and software. However, it contains useful information for application programmers, system managers, and system operators.

You are encouraged to read the technical summary selectively. Many of the system's concepts and features are repeated throughout the text in different contexts. You might first skim through the summary to find those topics that interest you most or perhaps read the summaries that appear at the beginning of each section. You can then start with sections of interest and know which section to refer to when you come across references to concepts discussed elsewhere.

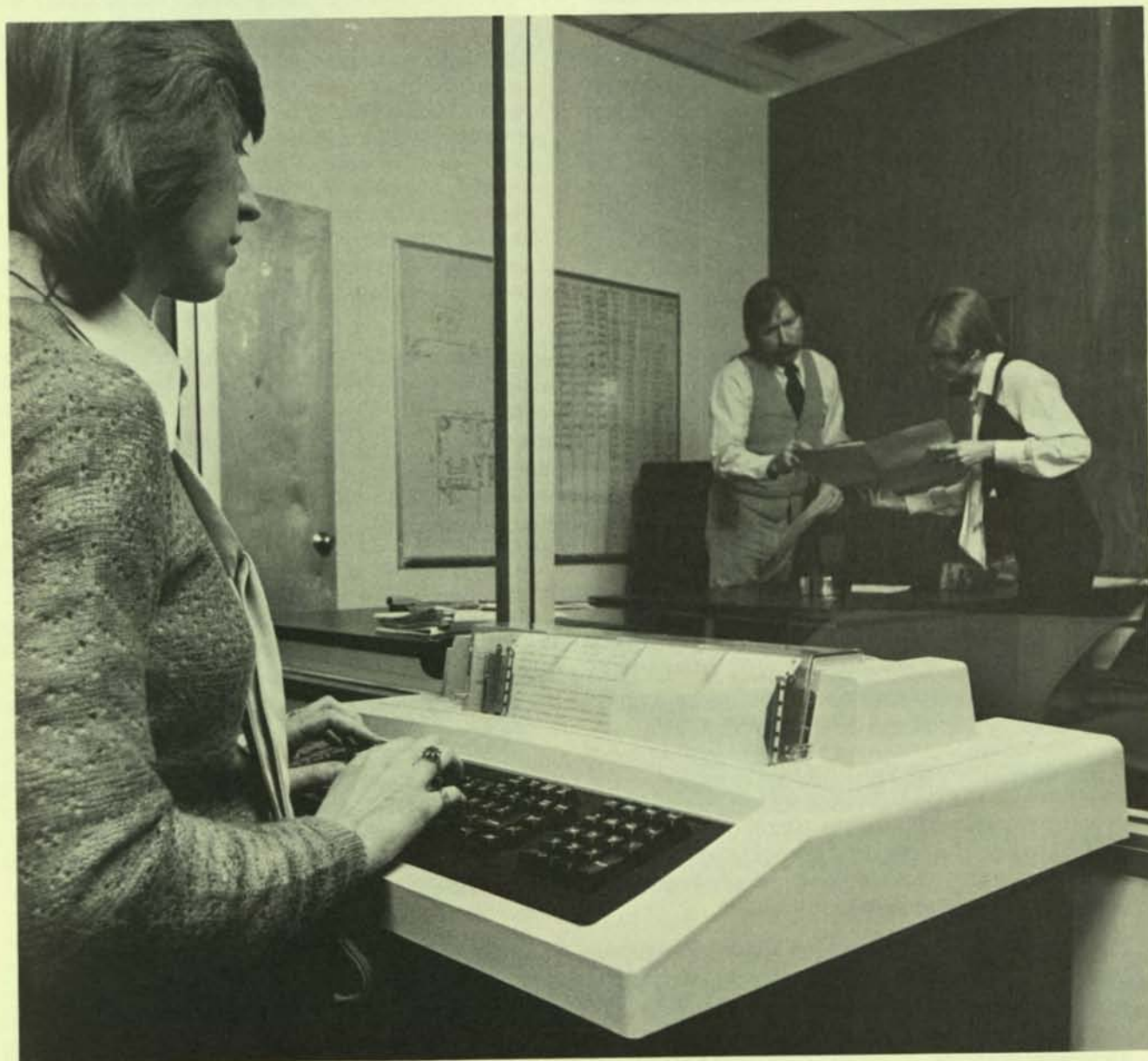
If you are familiar with computer industry terminology and simply want an overview of the DECSYSTEM-20 features, you should read Section 2, *Overview*. It briefly describes the DECSYSTEM-20 characteristics and introduces features of the system that are described in detail in the rest.

Some people may find it more helpful to begin with Section 3, *The Users*. It introduces many of the aspects of the system that support application programming, system programming, system management, and operator control.

If you are an application programmer, you will find that the sections on high-level languages, data management, and application products provide an indepth discussion on the system's characteristics and capabilities.

Finally, whether you are considering purchasing a system or have a system already, you should read Section 11, *Support Services*, to become familiar with the kinds of services DIGITAL makes available to its system users.

2 Overview



DECSYSTEM-20 computers are multipurpose systems that provide a wide range of computing versatility and power combined with exceptional reliability and efficiency. The systems have built-in protection mechanisms in both the hardware and software to help ensure a high degree of data integrity and system availability. Online diagnostics and error detecting and logging verify system integrity. Many hardware and software features provide rapid diagnosis and automatic recovery, should the power, hardware, or software fail.

The processors have a 36-bit architecture, and their instruction set is implemented in microcode. All instructions are capable of addressing up to 256K words of memory without resorting to base registers, displacement addressing, or indirect addressing. The large instruction set allows for the construction of efficient programs and access to a wide range of operating system functions.

The operating system that runs on the DECSYSTEM-20 series of computers is TOPS-20. Easy to use, TOPS-20 provides a highly reliable virtual memory, multipurpose operating environment. It supports multilanguage timesharing for program development and flexible forms of interactive and terminal-oriented applications, in addition to a full capability multistream batch system.

The operating system is both flexible and extendable. It enables the programmer to write large programs that can execute in both small and large physical memory configurations without the programmer having to define overlays or, later, modify the program to take advantage of additional memory.

DECSYSTEM-20 computers are state-of-the-art hardware and software systems. They serve interactive timesharing users, provide communications, and perform multiprogramming batch processing simultaneously. Because of the systems' versatility, they can be used for a variety of applications, such as:

- Commercial applications that require interactive data access, a variety of data types, online data editing, data integrity, and high throughput
- Scientific applications that require large storage capacities and high-speed computational capabilities
- Communications that require a large number of interactive terminals and fast response

The design of DECSYSTEM-20s easily accommodates expansion and the incorporation of new features, devices, and technology. Likewise, TOPS-20 is software-compatible in that programs that run on one version of TOPS-20 will operate on subsequent versions of TOPS-20 software without any program modifications.

DECSYSTEM-20s can supervise themselves primarily because of the hardware interrupt system and the TOPS-20 operating system. Their combination allows for rapid response to dynamic conditions both internal and external to the system. When an error occurs, the current context is saved, the interrupt is processed, and the saved context is restored, allowing the continuation of normal operation. There are multiple interrupt levels so that more important interrupts can interrupt less important ones. Also, there is a special set of fast accumulators used at interrupt level, that make it unnecessary to save and restore the accumulators associated with the running context.

A variety of input/output devices can be connected to DECSYSTEM-20 computers. The hardware features high-speed data channels, mass storage controllers for disk and magnetic tapes, and communications controllers.

COMPONENTS

The major components of the DECSYSTEM-20 are the:

- Processor — includes the choice of two models of CPUs that run the TOPS-20 software: the KS10 and KL10. Both processors use the same instruction set and support Metal-Oxide-Semiconductor (MOS) memory.
- Peripherals — include a range of disk drives with small and large capacity, magnetic tape systems, hardcopy and video terminals, lineprinters, cardreaders, and papertape reader/punch.
- Operating system — includes a virtual memory manager, a pager, a swapper, system utilities, device drivers, a file system, a command language, and operator and system manager tools.
- Languages — include TOPS-20 MACRO assembly language and, optionally, FORTRAN, COBOL-68, COBOL-74, BASIC-PLUS-2, ALGOL, APL, CPL, and BLISS-36. Development tools for both native- and compatibility-mode programs include editors, linkers, and debuggers.
- Communications — includes DECnet-20 software, IBM emulation and termination software for 2780/3780 and HASP stations, and ARPANET software.

Processors

The KL10 and KS10 central processing units (CPUs) form the basis of DECSYSTEM-20 computer systems. Both CPUs provide 36-bit addressing, eight sets of 16 fast general purpose registers, and seven priority interrupt levels. Both CPUs feature a microprogrammed instruction set capable of directly addressing up to 256K words of memory. TOPS-20 on the KS10 supports from 256K to 512K words of main memory, and, on the KL10, from 256K to 2,048K words of main memory.

Cache memory on both systems provides a faster effective memory access time. The cache implementation for the KS10 is 512 words and for KL10 is 2,048 words.

On the KL10, the PDP-11-based Console/Diagnostic front-end computer plays a key role in the operation and maintenance of the system. This minicomputer provides all console and command terminal functions, interfaces with the various peripheral devices, and performs all diagnostic and maintenance functions for TOPS-20 and the KL10.

On the KS10, the 8-bit microprocessor console is an extremely important subsystem because it performs all console functions and controls all diagnostic functions. Unit record equipment and communications are handled by a UNIBUS Adapter (UBA), which also controls the synchronous lines and the tape drives.

Both CPUs support KLINIK, the remote diagnosis capability that equips Field Service Engineers to examine the system while normal system operation continues.

TOPS-20 Operating System

TOPS-20 is a virtual memory operating system designed for many applications, including computation, data processing, transaction processing, program development, and batch processing. Some of the features that contribute to TOPS-20's versatility are:

- a multiuser, multiprogramming environment;
- interactive processing that allows for simultaneous sharing of system facilities by many users;
- multiple high-level language processors, two easily used but powerful text editors, and many system utilities;
- a multistream batch system;
- an easily learned and used command language which is common to both the interactive and batch users;
- online help facilities for commands and programs;
- a multiprocess job structure with a virtual memory address space of 256K words for all processes;
- a sophisticated, reliable, hierarchical file system that includes file sharing, file protection, file mapping, and file backup facilities.

TOPS-20 provides a command language that is interactive, comprehensive, easy to use, and extremely flexible. It enables the user to log into the system, create and manipulate files, develop and test programs, and obtain system information.

The interactive user who is not familiar with TOPS-20 can type a "?" (question mark) at any point within a TOPS-20 command to request help. For example, if a user types a "C" followed by a question mark, the command language processor lists all TOPS-20 commands beginning with the letter "C." It then reprints the line up to the question mark so that the user can continue, without starting over, with the desired command.

The user can give a abbreviated portion of a command and then press the "ESCAPE" control key on the terminal. If the abbreviation is unique, the command language processor will display the remainder of the command along with guide words to prompt the user for the necessary arguments. This feature, called Command Recognition, also works on command arguments, on file specifications, and on user names.

TOPS-20 also provides the user with help facilities that can be invoked simply by typing "help," followed by the program name or command name in question.

TOPS-20 provides a flexible, easy-to-use, powerful batch processing system that includes input spoolers, batch stream controllers, a queue manager, a task scheduler, and output spoolers. Batch jobs can be initiated and monitored from any terminal or can be automatically submitted by the system at specified times.

Both interactive users and batch users work with the same command language. This feature is important because it facilitates the use of either mode of processing. Equally significant is that time is saved, since users need learn only one set of commands for both interactive and batch processing.

The operating system is organized around a relatively small amount of directly accessible main storage and a large amount of high-speed secondary storage. All programs are organized in pages of 512 words. Such organization, which is transparent to each program, means users need not concern themselves with how memory will be allocated in main storage or how pages will be swapped between main and secondary storage. Assisted by the hardware, the operating system transfers pages between main storage and secondary storage as required, without explicit intervention on the part of the program or user.

The operating system provides a demand-paged memory environment. Every job on the system can be considered as a set of processes. These separately schedulable processes are linked together by user options or system-defined linkages, for efficient use of storage and system resources. Operating in a demand-paged mode, the system moves into memory a subset of all pages needed for each active process, bringing pages in from — or moving pages to — the disk as required. The system provides for easy, efficient communication among processes, both at the command and system level.

With TOPS-20, processes are able to initiate, suspend, and communicate with other processes. This structure allows for modularity in program development that can be taken advantage of at execution time by segmenting programs into distinct phases. By using a separate process

for each phase, programmers can overlap the phases of the program and thereby decrease the execution time of the overall program.

TOPS-20 uses this process structure in the implementation of its own supporting software. As an example, the command language processor is a process within the user's address space. Not only is the operating system smaller, because the command language is not part of it, but this design allows a job's command processor to control a process which is itself another command processor. New or special command language processors can be written, tested, and used during normal system operation, just like any other system program. Of course, for the user who does not require such a process structure, its support by the system is invisible.

Both file space and process space are integrated. Users having sophisticated needs, such as those with multiuser database applications, can simultaneously and efficiently access and update files, fully assured they are working with the latest data. Unlike other systems, when a TOPS-20 file is properly updated, it is the file itself, not a copy, that is modified.

The TOPS-20 file system is a general purpose, named file system that provides mountable disk structures, multilevel user directories, optimized file placement and I/O, and a file archiving and retrieval capability that's under user and operator control.

The TOPS-20 file system has been designed to provide controlled sharing of programs and data files. System efficiency is maintained by allowing for several users to simultaneously access a single copy of any program. However, TOPS-20 has built-in mechanisms to protect the system and the users from the errors of other users. If a user is reading a file or executing a shared program, and if another user attempts to write in one of the shared pages, the user doing the writing is given a private copy of the page and proceeds with modifying the page. This feature, called "copy-on-write," is automatically handled by the system. TOPS-20 provides the file facilities that equip users to create, access, and maintain fully protected data files. The flexible file facilities enable a wide variety of file organizations to be used, including sequential, relative, indexed sequential, and database organizations.

TOPS-20 support for many optional high-level programming languages provides flexibility in choosing the most effective or most familiar language for use in a given application. To minimize memory requirements, all language processors are sharable and generate sharable code. Job control, command languages, and the language processors are compatible under timesharing and batch processing.

TOPS-20 optionally provides three types of communications software:

- DECnet-20, which extends the facilities of TOPS-20 so that processes can communicate with other processes in a DIGITAL computer network;
- IBM Emulation/Termination, which performs both emulation and termination of IBM 2780/3780 remote stations or HASP workstations;

- TOPS-20AN, which allows a user to communicate with other sites or users on the Advance Research Project Agency Network (ARPANET).

RELIABILITY

DECSYSTEM-20 computers provide features needed for maximum system reliability. Built-in reliability for both hardware and software provides data integrity; increased uptime; and fast system recovery from power, hardware, or software failures. Some of the features that contribute to this reliability are discussed below.

Software Design

TOPS-20 is designed to easily and efficiently serve many users and, at the same time, to be protected from possible user errors. For example, the TOPS-20 command language processor is the controlling process in each user's job. This organization makes the overall system more reliable because a command error will affect only the job that caused the error; other jobs continue processing in the normal manner.

High reliability and security are necessary attributes of a multiuser environment, and the sophisticated TOPS-20 file system provides both. The file system allows controlled sharing and protection among users. Master file directories are redundant for increased file system integrity. File read operations have priority over file write operations, so the system can quickly satisfy page faults. File write operations are ordered to ensure that files are correct and remain in a consistent state. When operations on a file are active, index blocks and pointers are kept in memory for speed. When the user is notified that a file is closed, it has been completely written to the disk.

The operating system's division of nonresident and resident portions allows for efficient use of memory and for system functions to be logically divided for increased reliability. Unique protective processor modes are employed to effectively manage memory.

System Availability

The TOPS-20 operating system allows the hardware system to continue running even if some of the hardware components have failed. The system automatically determines the presence of peripheral devices on the system when the system is started. If the usual system bootstrap device is unavailable, the system can be bootstrapped from another disk drive, from magnetic tape, or from a floppy disk. If memory units are defective, memory is reconfigured so that defective modules will not be referenced. Software spooling allows for output generation, even if the normal output devices are not available.

The system operator can perform software maintenance activities without having to bring the system down for stand-alone use. Concurrent with normal activities, the operator can back up disks and run restoration procedures for all files on the system or just a single file.

The operating system supports online peripheral diagnostics. TOPS-20 logs CPU errors, memory errors, peripheral errors, and software failures. The operator or Field Service engineer can examine and analyze the error log file while the system is in operation.

System Recovery

Automatic system restart facilities bring up the system without need of an operator after a system failure caused by a power interruption, a machine check hardware malfunction, or a fatal software error. TOPS-20 automatically performs machine checks and internal software consistency checks during system operation.

Using TOPS-20 remote diagnosis, DIGITAL Field Service engineers can run diagnostics, examine memory locations, and diagnose problems from a remote terminal. The Field Service engineer who goes to the site is prepared in advance to correct the specific problem.

Error Reporting

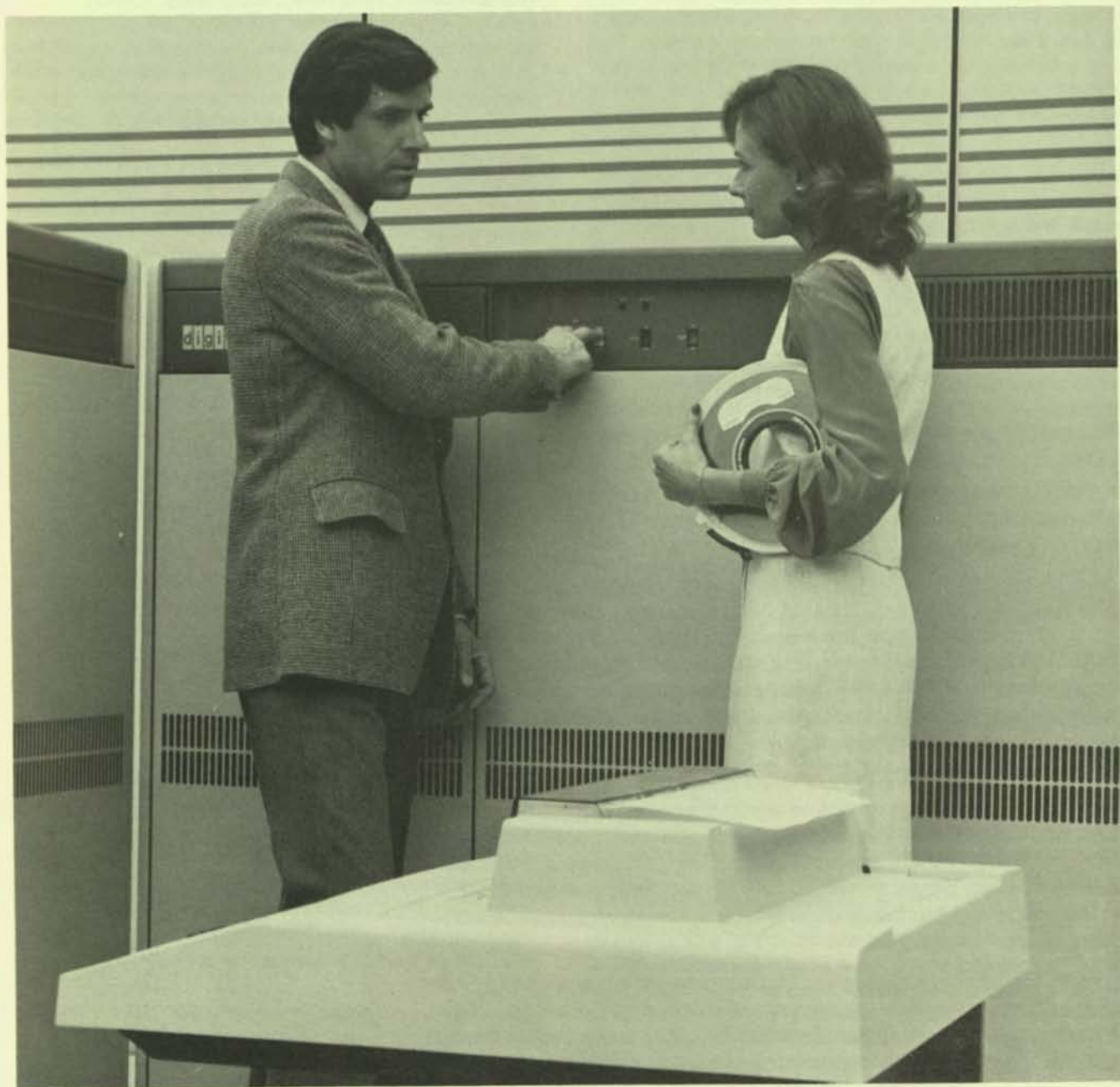
TOPS-20 provides an extensive error-detection and error-recovery package to ensure maximum system availability to the user. It also provides complete error reporting for DIGITAL Field Service and Software Support personnel and the customer's operations staff.

When an error is detected, TOPS-20 gathers all pertinent hardware and software information, including whether the error is recoverable. It then invokes the recovery procedure and adds this information to a disk file for storage. Later, a program can read this file and generate reports concerning the entire system or individual components. Additionally, significant operational events, such as system reloads and changes in the system configuration, are recorded to assist the operations staff in monitoring system performance.

Periodically, Field Service personnel gather summary information from this error file and use it to detect indications of possible problems. Detailed reports about specific errors frequently allow diagnosis of a problem and eliminate the need for running exhaustive diagnostics in a stand-alone mode.

The error file can be backed up on magnetic tape to provide a complete history of system operation and quickly pinpoint slowly degrading portions of the system long before serious problems develop.

3 The Users



The DECSYSTEM-20 consists of hardware and software that equip users to run a variety of programs efficiently and conveniently. The system is designed to execute many different kinds of jobs concurrently.

The DECSYSTEM-20 provides a complete program development environment. In addition to the native-mode assembly language, it offers optional high-level programming languages commonly used in developing scientific and commercial applications. It also offers the BLISS-36 implementation language for system programming applications. The DECSYSTEM-20 provides the tools necessary to write, assemble or compile, and link programs, as well as build libraries of source and object modules.

The system's general users are those who interact with applications or system programs from an online terminal or who benefit from executing batch jobs. These users can control the operation of the system through the command language. This language also can be used by system programmers to develop application software, by operators to monitor the system, and by system managers to control resources of the system.

The TOPS-20 operating system takes maximum advantage of system throughput capabilities, allowing many independent users to simultaneously share system facilities. Its rapid-response nature makes it particularly well suited for a wide range of tasks.

This section looks at the system from four different user perspectives:

- Application programmers
- System programmers
- System managers
- System operators

THE APPLICATION PROGRAMMER

The application programmer can write, compile, and test programs both interactively and in batch mode. As an interactive user, the application programmer can control the running of a program; create, edit, and delete data files; compile, execute, and debug programs; request assignment of peripheral devices, such as magnetic tapes; and make use of all system features from a terminal. As a batch user, the application programmer can submit a job into the system, using a card deck that contains control cards defining the command options and error-recovery procedures for the job or, using a terminal, the user can create and submit a control file that is then interpreted by the batch system and processed in exactly the same manner as the job submitted on cards.

Programmers can use the system for development even while other user jobs are in progress. All DECSYSTEM-20 languages and compilers are shared and reentrant, so many programmers can develop programs in the same language or in different languages without overburdening the system.

The Programming Languages

Languages offered on the DECSYSTEM-20 are:

- COBOL, the recognized "big machine" business language. DECSYSTEM-20 COBOL provides extensive data processing capabilities for commercial applications. It is written to the ANSI 68 or 74 specification, so it matches the COBOL used on most general purpose systems.
- FORTRAN, the standard scientific language. DECSYSTEM-20 FORTRAN consists of a globally optimizing compiler and runtime system with an interactive debugger to provide fast program development and execution.
- ALGOL, a sophisticated scientific programming language. DECSYSTEM-20 ALGOL is a one-pass, single-phase compiler that generates optimized object code.
- BASIC-PLUS-2, an easy-to-learn conversational programming language. BASIC-PLUS-2 uses familiar English words, abbreviations, and mathematical symbols to perform operations. Convenient data structures, including multi-key indexed files, can be easily created and efficiently accessed with BASIC-PLUS-2.
- APL, a concise programming language suitable for manipulating numeric and character-oriented, array-structured data. It includes procedural operators for array calculations and its own editing and debugging facilities.
- CPL, an interpreter supporting a subset of the ANSI PL/I language.
- BLISS-36, DIGITAL's implementation language for software development. It contains many of the features of a high-level language, yet it also provides the flexibility and access to hardware of assembly language.

Application Tools

Some of the tools available to the application programmer are:

Record and File Handling — The record- and file-handling services provide general record and file-handling capabilities to users of high-level languages. Programmers can include statements in their programs that can read, write, find, delete, and update records within files. Records can be of fixed or variable length. The programmer can choose the best file organization and record access methods for the data processing application.

For additional information on record and file handling and the system's data management techniques, refer to the section on Data Management and Application Products.

Database Management — DBMS is DIGITAL's proven CODASYL-standard database management system. DBMS enables programmers and managers to organize and maintain data in efficient structures, so it can be quickly and easily accessed by users and application programs. DBMS is an optional software product. Refer to Section 9 for a detailed explanation of the DBMS software product.

Transaction Processing — The interactive transaction processing capabilities of TOPS-20 are supported by TRAFFIC-20, DIGITAL's Transaction Routing and Forms Formatting in COBOL.

TRAFFIC-20 simplifies building easy-to-use transaction processing COBOL applications that let the computer guide the user through each transaction. It is not necessary for the user to learn system commands. The application can be written so that the user simply fills out a form on the terminal screen. In effect, TRAFFIC-20 makes nonintelligent terminals appear to be intelligent terminals that can display a form or a series of forms.

Used with the DBMS system, TRAFFIC-20 makes it easy to build flexible, integrated, multiuser applications that include specialized screen formats for data input. Because the DBMS system is integrated into COBOL, several transactions can access and update files at the same time. Data integrity is ensured. Centralized data definition is supported.

THE SYSTEM PROGRAMMER

The system programmer can design and develop application systems for multiprogramming environments requiring fast response and a high degree of job interaction and data sharing.

System programmers are provided with a range of tools for developing high-level languages, system utilities, and application systems where the sophisticated features of the operating system are required.

MACRO, the TOPS-20 symbolic assembly language, makes machine language programming easier and faster by translating the symbolic operation codes in the source program into binary machine language instructions. It relates symbols specified by the user to stored numeric values and it assigns relative core addresses to symbolic addresses of program instructions and data.

System programmers can also use BLISS-36 for software development. BLISS-36 is an optimizing, high-level system implementation language for TOPS-20. It is specifically designed for building compilers, system utilities, and operating system software.

System programmer applications normally communicate with the system through monitor calls. Monitor calls are used to request system functions, such as input/output operations, error handling, and number conversions, during execution of the program. Support programs, such as the COBOL compiler, issue requests to the system through monitor calls. All monitor calls are reentrant, to maximize system reliability by isolating monitor-call processing from operating-system processing.

Through TOPS-20 monitor calls, users can access a variety of sophisticated operating system functions — for example, the TOPS-20 command processor. Applications can be written that have the same syntax, help facilities, and recognition features used at system level.



THE SYSTEM MANAGER

The system manager is responsible for planning data access and protection, granting privileges, authorizing use of the system, controlling resource utilization, and analyzing the system's accounting and performance information.

User Authorization

The system manager controls use of the system primarily by creating user directories that identify the user, supply defaults, specify privileges, and limit resource usage.

Each user who wishes to access the system is assigned a user name, password, and account name. When the user logs into the system, the system verifies the user's name and password. If either is incorrect, the system refuses the user access, thereby protecting itself from unauthorized use.

The system manager can assign a default account name in the user's directory so that the user does not have to enter an account name when logging into the system.

Privileges

The system manager can assign individual users special privileges that cover directory access, resource usage, and system operational control.

If a user tries to execute a function that requires a specific privilege, the system checks the user's directory parameters to see if the user is allowed that privilege. If the user does not have the specific privilege, the system does not execute the function and instead notifies the user of denial of that privilege.

Allocating Disk Storage Quotas

The system manager assigns each directory a specific number of pages for both working storage and permanent storage. Working storage refers to the disk space that a user can have while working on the system. Permanent storage refers to the total disk space in which a user can store files after logging off the system.

The system strictly enforces working storage allocation. It will not allow a user to create a new file if that individual's working storage has been filled.

Controlling Resources

Through a user-written program, the system manager can make policy decisions that govern access to a specific system resource. For example, TOPS-20 allows a user to change the speed of a terminal, assign a device, log into the system at any time, mount a magnetic tape, and mount a disk structure. Access control enables the system manager to restrict or disallow the use of some of these facilities. The system manager can grant certain users access to the system at specified times of the day and, if desired, at specified terminals. By using the access control mechanism, the system manager can reduce or prevent malicious access to the system resources and has an additional means for collecting accounting or other information.

Scheduler Controls

The system manager is provided tools that enable control or tuning of the allocation of CPU time. The tuning mechanisms include class scheduling and bias control.

Using class scheduling, the system manager can allocate percentages of the central processor's time to individual classes of users to provide consistent service. Each job in a class receives a portion of the class percentage. The system manager can also set up a class to include all batch jobs, thereby controlling batch jobs separately from interactive jobs.

Class scheduling is dynamic, and changes made during a session remain in effect until the system is reloaded or another change is made. Using TOPS-20 commands, the system manager can turn the class scheduler off and back on again while the system is in operation.

By using the bias control feature, the system manager can direct the scheduler to favor either interactive or compute-bound programs. The bias control feature is analogous to turning a knob over a range of settings. When a low number is selected, the scheduler favors users running interactive programs. When a high number is selected, it favors users running compute-bound programs.

Bias control and class scheduling can be used simultaneously.

Resource Accounting Statistics

Computer use can be assigned and charged to various accounts via the accounting facility. Using this feature, the system manager can add security to the system; determine charges for computer usage and bill users by account name; and associate classes with accounts for use by the class scheduler.

All accounting data is stored in a usage file that can be accessed for reports and billing. Because the system collects detailed data in record format, the system manager can define algorithms used in resource usage billing.

Performance Analysis Statistics

The Performance Analysis Statistics program collects system usage and performance data that can be used to tune the system. Some of the statistics gathered by the program are:

- *Monitor statistics* indicating CPU, disk, memory utilization, and system performance;
- *Job statistics* indicating CPU time used by each job, information on each job's use of system resources, and statistics collected on the individual jobs;
- *System utilization statistics* indicating the demands made on the system and the distribution of system resources;
- *Disk I/O statistics* indicating the number of seeks, reads, and writes performed by each disk drive.

THE SYSTEM OPERATOR

The system operator has complete control of the system. Operator responsibilities include readying the system for timesharing and batch work, responding quickly to user requests such as magnetic tape mounts, taking care of the peripheral devices and the distribution of output, and recovering the system when error conditions occur. The system operator can be called upon by the system manager to perform a variety of other functions, related to both hardware and software, to ensure the efficient running of the system.

Operator Interface

The OPR unified system interface enables the system operator to deal with all the batch system components, handle user requests, and monitor application control programs. The operator learns a single set of commands for

operating the entire system and is provided with command recognition, guide words, and help text.

Controlling Batch Streams

The system operator can control the number of batch streams that can run concurrently. Batch jobs can be submitted by interactive users, other batch jobs, or programs. When the number of jobs exceeds the number of streams, the overflow is held in a batch input queue. The operator can start additional batch streams, if desirable, and can also stop, continue, and shut down batch streams. The operator can send messages to batch streams, set parameters for batch streams, and display the status and parameters of batch streams.

System Recovery

The system operator can select manual or automatic system recovery following a power interruption or a hardware or software failure.

On automatic system recovery, after a power interruption, the system determines whether the contents of memory are still valid. If they are, it restarts all possible I/O in progress at the time of interruption and continues operation from the point of interruption. If the contents of memory are not valid, the system automatically restarts itself from disk and executes startup command procedures.

If the standard system bootstrap device is unavailable, the operator can boot the system from one of several different devices.

System Backup

All files or selected files can be saved on tape. The system operator can use this tape to restore all or certain files back on to the disk.

If the system is using the archiving/migration feature, the operator can periodically run a program to copy all archived or migrated files to tape.

USER UTILITIES

The TOPS-20 operating system features several utility programs that aid the user in performing various tasks. The most frequently used utilities are:

- A MAIL program with which the user can send messages to other users. Messages sent by the MAIL program are stored in the receiver's disk directory and can be read at the receiver's convenience.
- A RDMAIL program to read the messages sent by the MAIL program.
- A PLEASE program by which the user can communicate with the system operator.
- A CREF program to generate a cross-reference listing of symbols used in programs.
- A FILCOM program to compare two files on a line-by-line or word-by-word basis and list the differences.
- A MAKLIB program that manipulates, changes, and queries relocatable binary files. MAKLIB is mainly used to manipulate and optimize collections of binary programs into libraries.

- A DUMPER program that lets the user copy disk files to magnetic tape for safekeeping and/or to transfer files between systems. DUMPER can be used by the system operator to perform full or incremental system backups. It can also be used by system users to back up selected files.
- A LINK program to merge independently compiled program modules and system modules into a single module that can be executed by the operating system.

- A DDT program that lets the user examine, search, change, insert breakpoint instructions, and stop/trace a program at symbolic level. The user can also single-step executable programs.
- An EDIT program with which the user can create and modify line-oriented programs and data files online.
- A TV program with which the user can create and edit any source documents, programs, or other text files.

4 The Operating System



TOPS-20 is a highly reliable operating system designed to provide a wide range of services to users of varying capabilities. After brief training in using the command language, even novices can easily use the system. The same command language serves both interactive and batch users.

TOPS-20 provides a demand-paged, process-structured environment. Every job on the system is made up of one or more processes, each having its own address space. Operating in a demand-paged mode, TOPS-20 moves into memory a subset of all pages needed for each active process and moves onto disk all pages not currently required.

The virtual memory design of TOPS-20 minimizes the use of main memory because only the currently necessary functions and data reside in memory. User programs are handled in the same manner, with nonactive pages residing on disk. Pages are moved into memory when needed, so the program can execute whenever the processor calls for it. This dynamic allocation does not require user intervention; it is jointly handled by the hardware and the software.

TOPS-20 is designed for interactive processing and for many independent users simultaneously to share system facilities. TOPS-20's rapid response and versatility make it particularly effective over a wide range of tasks. Most users communicate with the system through a terminal, either at the central computer site, or at remote locations connected to the system by communications lines.

THE COMMAND LANGUAGE PROCESSOR

The user interface to the system is the TOPS-20 Command Language Processor. It interprets all commands issued by the user for system services and sends responses from the operating system back to the user's terminal. The Command Language Processor, a user program residing in the user's address space, is used for:

- Accessing the system
- Directory and file utility operations
- Initiating programs
- Initiating operations
- Acquiring information about the system
- Debugging programs

Because the Command Language Processor runs as a user program, an error found by it will terminate only the job that caused the error.

DECSYSTEM-20 owners can purchase Command Language Processor source code. It can be modified, tested, and debugged as a normal user program under timesharing. The modified code can then replace the existing system version of the Command Language Processor.

The Command Language Processor permits a user to interrupt a currently executing program, save its address space, and start another program. For example, if a user neglects to create a necessary file before starting a program, the user can interrupt the program that is running, return to the TOPS-20 command level, and save the program's address space and execution state. After creating the necessary file, the user can return to the first program at the point of interruption and continue processing.

The Command Language Processor is reentrant and sharable. Even though it is shared by many users, only the currently needed parts of the Command Language Processor reside in memory.

The TOPS-20 Command Language Processor provides a common command language for interactive and batch users. The command language uses English words to invoke system functions. All TOPS-20 commands begin with a key word and can be followed by parameters to fully describe the function requested by the user. Users can create, edit, delete, and undelete files; compile, execute, and debug programs; and use all system features.

To enable even a beginner to use the system successfully, the command language contains such features as guide words that prompt the user, recognition input that uses command defaults to complete commands, and help facilities that provide instructions about programs and commands.

Simply by typing a single command, users can display all TOPS-20 commands or various system statistics — for example, terminal characteristics, system defaults, or available devices. Users can send messages to, or receive them from, other users; attach to another user's terminal; set terminal characteristics; and so forth.

THE BATCH SYSTEM

TOPS-20 can concurrently execute batch jobs and time-sharing jobs. The user can enter a job into the batch system by creating and submitting a control file from a terminal or by using a card deck. Because the command language for batch and timesharing is the same, the user can duplicate any terminal session with a batch job by using the same sequence of commands.

Switches provided with the batch system give the user great flexibility. Included in the command string, they can define the operation and set priorities and limits on memory and processor time. In addition, the user can structure the job submission so that jobs execute in a particular order.

Users can control the handling of error conditions by including special commands in their jobs. If an error occurs, and if the user has not specified any error-recovery procedures, the batch system initiates a memory dump of the user's area and terminates the job.

The batch system requires little or no operator intervention, although the operator can exercise control if it becomes necessary. The operator can specify system resources to be dedicated to batch processing; can limit the number of jobs, memory, and processor time for individual jobs; and can stop a job at any point, requeue it, or modify its priorities. The operator and the user can examine system queues to determine the status of all batch jobs. The batch system sends information to the operator and records on disk a log file of all messages printed on the operator's console. All operator intervention during the running of a batch job is recorded in both the user's log file and the operator's log file for later analysis.

Running several batch jobs concurrently enhances system and user throughput because a user can separate jobs into several parallel subjobs.

Batch system programs include the input spooler, the batch controller, the centralized queue manager, the task scheduler, and the output spooler.

The input spooler reads the input from a device, such as a cardreader, disk, or magnetic tape, and requests the queue manager to enter jobs into the batch controller's input queue. The input is separated according to the control commands it includes, and is placed either into the user's data files, or into the batch controller's control file for subsequent processing. The input spooler creates the user's log file and enters a report of its processing of the job as well as a record of any operator intervention during the processing. This log file is a standard part of the output for a batch job.

The centralized 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 according to external priorities, processing time limits, and parameters specified by the user for the job. The latter can include start and deadline time limits for program execution.

The output spooler improves system throughput by allowing the output from a job to be written to a disk for later transfer to the lineprinter. The queue manager places the job's log file and output into one or more output queues to await spooling. When the lineprinter becomes available, the lineprinter spooler processes the output. This spooler also has features such as special forms control and character sets, multiple copies, and accounting information. The output spooler and lineprinter spooler are also used by the interactive user to get a printed listing of a file.

JOBS AND PROCESSES

The TOPS-20 operating system distinguishes between a job and a process. When a user types a CONTROL C at a terminal, the system creates a job by invoking the TOPS-20 Command Language Processor. During the existence of the job, the user issues commands and executes programs. The system terminates the job when the user logs off the system.

Each job can have multiple, heirarchically organized processes. When a job is created, it consists of one process containing the Command Language Processor. Whenever the Command Language Processor cannot perform a requested function, it creates an inferior process to do so. The creating, or superior, process has complete control over its inferiors. It can give or withhold privileges, and suspend, continue, or terminate them. Processes communicate with each other using a Process Identification Number (PID) that is unique for each process in the system. In this tree structure, the top-level process in a job is usually the TOPS-20 Command Language Processor.

Each process is separately scheduled, has its own 256K address space, and competes for system resources such as the CPU and the devices. This makes the system more responsive to demands and it more easily achieves the goal of appearing to simultaneously perform a variety of functions.

Before a process can be scheduled by the system it must be in a runnable, or active, state. One requirement for this is that a subset of its pages needed for execution, called the working set, is in memory. The collection of processes most eligible to run and whose working sets simultaneously fit into memory is called the balance set. Processes that are not runnable are inactive because they are waiting for events such as terminal input. These inactive processes are blocked until the required event occurs.

Because the runnability of a process is event-driven it changes whenever the state of a process changes. To ensure good response, interactive processes tend to get higher priority than compute-bound processes. However, if a compute-bound process has been blocked for a long time, its eligibility is changed to allow it to run.

PROCESS COMMUNICATION

A process can communicate with other processes in the system in several ways:

- Direct process control
- Interprocess Communications facility
- Software interrupts
- ENQueue/DEQueue facility
- Memory sharing

Direct Process Control

A process can create and control other processes inferior to itself within a job structure. The superior process can begin, suspend, and resume execution of the inferior process. After the inferior process has completed its tasks, the superior process can delete it from the job structure.

Interprocess Communication Facility

Although each process is fundamentally independent, an important advantage of a multiprocess job structure is that the output of one process can be used as input to others. The Interprocess Communication Facility (IPCF) enables fast communication among processes. Communication occurs when processes send and receive information in the form of packets (messages). Each sender and receiver has a unique process identification number assigned to it that is used to route packets among processes.

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

To use IPCF, the system administrator must assign a user quotas that designate the number of sends and receives the user's process can have outstanding at any time. If a user has a send quota of two and has sent two messages, for example, the user cannot send any more until at least one message has been retrieved by its receiver.

Software Interrupt System

The software interrupt system enables a process to respond to several types of interrupts generated by other processes, such as hardware error conditions or terminal transactions. Interrupts can be activated either by a user or by 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 condition interrupts
- I/O error-condition interrupts
- Program errors (such as arithmetic overflow)
- Interprocess interrupts

For priority ordering, there are three interrupt priority levels assignable to each software interrupt channel. A system call is provided for interrupt dismissal to resume the interrupted code.

Enqueue/Dequeue (ENQ/DEQ)

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

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

Memory Sharing

Each page in a process' address space is either private to the process or shared with other processes. Pages are shared among processes when the same page is represented in more than one address space. This means that two or more processes can identify and use the same page of physical storage. Even when several processes have identified the same page, each process can have a different access to that page, such as access to read or write that page.

THE TOPS-20 MONITOR

The TOPS-20 monitor is a collection of modules that provides overall coordination and control of the total operating system. The monitor permits several user programs to be simultaneously loaded into memory. It makes optimum use of the timesharing hardware to prevent one user's program from interfering with other users' programs. A sophisticated scheduling routine handles the requirements for execution, memory, and system resources to provide good response for interactive users, and high throughput for computational users. The scheduler selects processes to run based on class, internally computed priorities, and external policies. It makes these selections frequently enough for all processes to seem to run simultaneously.

Another function of the monitor is to process input and output commands. The monitor's input/output service routines preprocess the data so that all devices appear identical to the user's program, thus simplifying coding. The monitor uses the software interrupt system to overlap input/output operations with computation operations. For instance, if a user's program must wait for completion of an input/output operation, the monitor will automatically



switch to another user's program, taking advantage of the system's throughput capabilities.

The TOPS-20 monitor is modular and paged. It can be tailored to each user's equipment configuration and application requirements. Extensions to the system in the form of peripheral devices and application programs can be added, altered, or deleted as required, without any change to the monitor.

In summary, the monitor exercises primary control of the routines that constitute the operating system. This makes the computer a flexible tool that provides the user with maximum use of the hardware's advanced design features.

VIRTUAL MEMORY

From a hardware view, memory is divided into sections of 256K locations. The paging hardware further divides each section into 512 locations. For the most part, virtual memory from TOPS-20's view is the same as from the hardware's view. TOPS-20 incorporates features that make virtual memory more appropriate for the end user. Users have a 256K word address space in which to place programs and data. TOPS-20 divides the user's space into 512 pages, each of which contains 512 36-bit words.

Virtual memory addressing is accomplished through the indexed access method. Section tables exist for each process, and page tables for each section. These page tables contain pointers to pages that contain either additional tables or the data. There are four types of pointers: no access, immediate, shared, and indirect.

The paging hardware maps pages from virtual address space into pages anywhere in physical memory. A page map for each process specifies correspondence from virtual address to physical address, whether an individual virtual page is accessible, whether it is public or concealed, and whether cache memory can be used to refer to the page.

VIRTUAL MEMORY OPERATIONS

The TOPS-20 operating system provides a demand-paged virtual memory environment. The system moves into memory a subset of all pages needed for all active processes, bringing pages in from or moving pages to the disk as memory space becomes available. The working set is that set of pages referenced during a recent interval of time, that is, those pages recently used. The system determines the interval so that, when the process is reactivated, memory references will most likely be confined to the same pages. A particular program's characteristics determine the number of pages in its working set.

A second definable property is the balance set, the set of processes most eligible to run and whose collective working sets fit into memory.

The working set manager uses balance set membership to determine whether a working set can be swapped into memory. Balance set membership is also used in calculating the history of a process.

The specific goal of working set management is to swap working sets into and out of memory based on working set histories and current memory allocation. To run, a process needs its working set in memory; therefore, the working set manager always attempts to have the working sets of runnable processes in memory. This aids throughput by keeping the CPU active with useful work. Some definitions are needed to understand the working set manager:

- **Preloading** — The action of swapping the working set of a process into memory upon the selection of the process to be run.
- **Postpurging** — The action of completely removing the working set of a process when the process has been selected to be swapped out to secondary storage. If postpurging is not in effect, the pages of the process are swapped out of memory one page at a time as other processes need the pages.

Part of the working set manager's decision to swap a working set in or out of memory is based on the current memory allocation. Membership in the balance set indicates that a process has been allocated memory. For this reason, the working set manager selects only a member of the balance set to be swapped into memory.

The other part of the working set manager's decision to swap a working set in or out of memory is based on a value representing the history of a process. This value is calculated from the current state of the process and its recent behavior. The state of a process includes whether it is runnable and its current queue location. The recent behavior of a process is related to its membership in the balance set.

The working set of a process is swapped into memory using one of two methods:

- It can have just its overhead pages swapped in. As the process executes, additional pages are faulted in as needed.
- The working set is preloaded. This method loads the pages that were in the working set when the process was last swapped out of memory.

The system manager decides whether to preload the working set. A working set is always postpurged when it is removed from memory. All pages of the process are swapped out of memory.

The Scheduler periodically redefines the balance set. Working set sizes are modified dynamically as a process runs and they are regularly monitored by the Scheduler to adjust balance set membership. If needed, a specific user or group of users can be guaranteed a percentage of the processor time.

TOPS-20 supports memory management through hardware and microcode. Memory mapping and page-level access protection provide efficient sharing of pages between processes and efficient context switching. Both page status and history for each page in memory are automatically updated by the microcode from information initially set up by the software. These updates create hardware tables that minimize overhead while they maximize system information needed for management of memory and disk-channel traffic.

To effectively manage memory, the DECSYSTEM-20 uses a combination of registers and tables to relate program or virtual addresses to physical addresses. Two hardware registers in the central processor, the EBR and UBR, contain pointers to the physical pages in memory that contain the mapping information for the operating system and the currently active user. These pages, called 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 the operating system. Three types of pointers contained in the process tables are immediate, shared, and indirect. The characteristics of an Immediate Mode Pointer are shown below. Entries in the User Page Table point directly to physical pages in memory. Shared Mode Pointer characteristics are also illustrated.

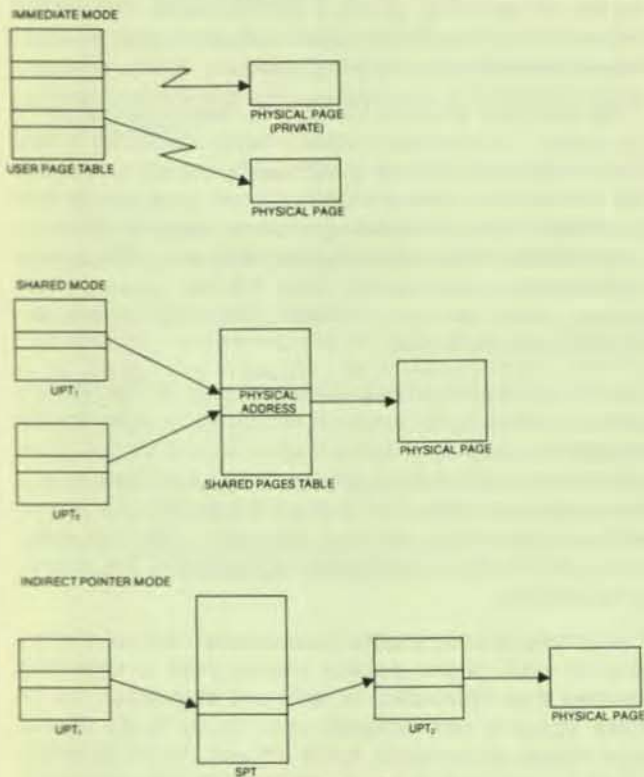


Figure 4-1
Pointer Modes

The shared-mode pointer indexes into a Shared Pages Table that contains entries pointing 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 Pages Table is that, 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 Pages Table. Without a Shared Pages Table, every user's Page Table, many of which would not be in memory, would have to be modified. When one process writes on a shared page, the system performs a *copy-on-write*, which creates a process-specific copy of the page, including the modifications. Because the process is using a modified page, and no longer sharing the original page, the Shared Pages Table is updated to show one less user sharing that page.

In indirect mode, the entry on the Shared Pages Tables does not point directly to memory, but rather to another entry in the Process Table. This other entry may reindex back into the Shared Pages Table or point directly to a physical page.

To maximize system speed, the Hardware Page Table in the CPU includes a subset of the information for all active process page mappings, including the operating system. 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 references to these pages then proceed through the Hardware Page Table, eliminating the need to access the in-memory page tables. This high-speed Hardware Page Table minimizes the need for an extra cycle to pick up the required mapping information.

An additional register in the central processor points to a memory status table that contains, for each page of physical memory, information on recent references, page modifications, reasons for a page to be loaded in memory, and reverse pointer data. The operating system uses the memory status table to compute the length of time pages have been active and in memory, to record when pages have been written into, to record which processes have referenced the page, and to locate which of the page tables point to the page.

THE FILE SYSTEM

The TOPS-20 file system is designed to provide the user with the highest possible degree of operational flexibility in storing and retrieving data. Because the system adjusts the interface between the file and the device, the user need not consider the physical characteristics of the recording device when creating most files. The system invokes security measures to help ensure that files are not subject to unauthorized use or destruction. Users have facilities to back up files from mass storage devices to magnetic tapes, as well as to restore such files to the mass storage devices when they are later needed.

Files

A file is an organized collection of information directed toward a certain purpose. The TOPS-20 file system is a collection of named files composed of 512-word data pages. Each file is identified by a node name, device, directory name, filename, filetype, and generation number — all of which define a unique path to the file.

File Protection

A user who creates a file can either assign a protection code or let a system default protection be assigned. 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 all, some, or none of the following:

- Acknowledgement that a file exists when listing the directory
- Permission to append data to a file
- Execute-only access
- Write access
- Read access

Each file is associated with an information block that fully describes the file characteristics to the operating system. The information includes the file protection code, the owner of the file, the name of the user who last wrote into the file, the date the file was created, the date the file was last changed, and the size of the file. The owner of a file can change the protection code of the file at any time. All user filenames are stored in a user file directory.

File Directories

The TOPS-20 operating system maintains a file directory for each user. The file directory contains the names of all the user's files, contains pointers to the user's file information block, and contains general user information that includes the user's password, privileges, disk-space allocation, disk space used, and default protection code. All file directories are themselves named files, and information about them is contained in a master directory called ROOT-DIRECTORY. There are two copies of ROOT-DIRECTORY to increase file-system integrity.

The tree-structured directory facility allows the user to create subdirectories. Subdirectories appear to the system as do file directories in that they can be accessed, can be members of user directory groups, and have the same protection mechanisms as file directories. The tree-structured directory facility permits approximately 4,000 directories to be created on a KS10 system and 12,000 directories on a KL10 system.

Groups

A group is a set of cooperating users that is established by the system manager. Each directory contains two membership lists: the list of user groups of which the owner of the directory is a member and the lists of groups of which the directory itself is a member.

File Usage

The prime concern for users of the file system is the mechanism that permits controlled sharing of programs and data. The design of TOPS-20 ensures that two users with access to the same program will share it automatically. When either user modifies a page of the program, the system gives that user a private copy of the page, unless the two users explicitly specify they wish to share the modified page. This facility makes memory utilization far more efficient, since many users can share a single physical copy of any program. For example, although the TOPS-20 COBOL compiler code is shared by all users, the users' particular program statements are stored in separate areas.

Although program sharing is important, data sharing is even more important when it comes to building effective multiuser database systems, especially when online interaction is required. It can be confusing if one user is changing a particular file while another user is reading it.

To prevent this confusion the system offers several modes of shared access:

- Shared Reading (one or more users reading, no user writing).
- Writing (one user writing, none reading).
- Shared Updating (one or more users reading and writ-

ing simultaneously, using the TOPS-20 EN-Queue/DEQueue facility for coordination).

- Restricted Updating (one user writing, any number of users reading).

As a further protection, if a user has opened a file for reading or is executing a shared program and attempts to modify a page that other users believe is not changing, the user who wishes to modify the page is given a private copy of it. This automatic facility is called *copy-on-write*.

Archiving Files

The number of files in a user's directory is constantly growing. Consequently, the user may want to delete some files or store some offline on magnetic tape. TOPS-20 users can store files offline with a simple command. The user requests a file to be archived, and the system marks the file for archiving, giving it archive status. The system may then make the file invisible to the user, even though it has not yet been physically archived.

A file becomes archived when it has been backed up on two tapes. The two tape copies provide insurance in case one of the tapes is lost or destroyed by accident. An archival run consists of executing a system program to write onto tape files with pending archival requests. Thus, to fully archive a file requires two archival runs. The system administrator must decide when the two runs are performed, either one after another or the first run one day and the second the next.

An archival run is performed in two phases. During the first phase, all files in the range of the given file specification are examined to see if the archival request bit is set. When a file is found with this bit set, the system proceeds to write the file to tape. After the file is written to tape, the search continues for more archival requests. After the entire range of the file specification is searched, the second phase begins.

The second phase consists of examining the tape information for each of the archival request files to determine whether that information is valid and complete. The archive status is then adjusted accordingly. If the file has been written to two tapes, and if the user did not specify to retain the contents of the file on disk, the disk contents of the file are deleted. Because the filenames are retained in the user's directory, the user can request retrieval of the files by filename. The file retrieval process uses the information in the file directory to identify the tape reel containing the file.

Tape Labelling

The TOPS-20 operating system supports the reading and writing of ANSI- and TOPS-20-labelled tapes, and the reading of EBCDIC-labeled tapes. This support includes Automatic Volume Recognition (AVR) of labeled tapes; reading and writing of labels; automatic record blocking and file searching; handling multiple-volume tape sets; and fixed, variable, and spanned record formats.

CONSOLE FRONT-END PROCESSOR

On the DECSYSTEM-2040 and 2060, the console front-end processor and software reduce the central proces-

sor's participation in I/O operations and together serve as a powerful diagnostic/maintenance tool for service personnel. Specifically, they are responsible for providing:

- Console functions
- Interface for command terminals
- Interface for unit record peripherals
- Diagnostic and maintenance functions

Console Functions

Two major functions of the front end are system initialization and communication between the system and the operator. A person need only push a button and type in data to initiate the automatic program sequence in which the console 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 can request a memory configuration listing that indicates which memory controller is online, what the highest memory address configured is, and how the memory is interleaved.

All normal console capabilities, such as the ability to examine registers, change registers, start, stop, reset, and load, are provided by the console front-end processor. An operator command language is provided with the system.

Command Terminal Functions

The console front-end processor serves as an intelligent data link and buffer for the interactive terminals, relieving the central processor of this overhead. The console front-end processor buffers the characters received and interrupts the central processor, either upon receipt of a clock signal or in the event it has a group of characters to send. The central processor also sends groups of characters for terminal output to the console front-end processor, further enhancing efficiency. Programmable terminal speed-set-

ting capability is provided, enabling terminal speeds to be changed dynamically. On dialup terminal lines, the monitor will automatically select the appropriate rate for 110, 150, 300, or 1,200 baud lines when the user types a carriage return or Control-C. Thus, line speeds need not be associated with phone numbers.

Peripheral Interface

The unit record-spooling programs in TOPS-20 communicate with the console front-end processor, using the same buffering mechanism described for the command terminal functions. Both the central processor and the console front-end processor maintain buffers for data. They interrupt only once per buffer transmission, increasing the amount of useful work each processor can do.

Diagnostic/Maintenance Functions

Remote diagnosis capability reduces mean time to repair (MTTR) and increases system availability to the user. It also allows DIGITAL's service engineers to determine the causes of most system failures before leaving the local office. By running diagnostic tests using telephone dialup facilities, the Field Service engineer can give the customer better service, because it is easier to select which spares and test equipment should be taken on the call. Certain tests can be run during general timesharing. Even if the main CPU is inoperative, the console front-end processor can access all buses and major registers.

Total system security is maintained because the onsite system operator must activate the communications link. Only the onsite personnel know the specific password to the system each time remote diagnosis is employed. Time limits for system access can also be imposed, and the remote link cannot operate at a privilege level higher than that specified by the local console terminal. All I/O to the diagnostic link is copied to the local terminal, giving onsite personnel a record of all steps taken.

5 KL10-E Processor



The KL10 central processing unit is the heart of DIGITAL's large-scale computer systems. It provides the basis for the popular and successful DECSYSTEM-10 and DECSYSTEM-20 computer families.

The KL10-E version of the KL10 is the central processing unit of the DECSYSTEM-2040 and the DECSYSTEM-2060.

Important features of the KL10-E central processing unit are:

- 36-bit word length for programming efficiency and data resolution
- 398 logically grouped, microprogrammed instructions including byte and string manipulation, double-precision fixed and floating point, and character editing and conversion
- Full 256K, 36-bit word memory space addressing without indirect or base addressing
- An architecture that allows modular expansion without component replacement or complex reconfiguration
- An integrated PDP-11 front-end processor for diagnostic functions, console functions, and handling of low-speed peripheral devices
- Extensive error-detecting circuitry for maintainability and availability
- Integrated high-speed data channels/controllers for disks and tapes for improved reliability and price/performance
- State-of-the-art memory management hardware for virtual memory operation
- Four-word fetch capability for increased bandwidth and performance
- High-speed cache memory with state-of-the-art design; typical inline programming results in a 90 percent cache "hit" rate for efficiency and high performance
- One of the most flexible priority interrupt systems available
- A full line of high-efficiency, high-performance UNIBUS and MASSBUS peripheral devices to meet individual user needs
- High-speed ECL logic implementation
- Eight sets of general purpose registers that can be used as accumulators, index registers, or the first 16 locations of main memory, having an access time of 125 nanoseconds

The internal architecture of the KL10-E central processing unit can be considered a collection of special purpose processors connected through high-speed internal buses. Four major subsystems constitute the central processor for this discussion:

- The Execution Box (EBox) and associated buses that execute machine instructions and provide control for other subsystems
- The main memory subsystem that consists of a memory control unit (MBox), associated buses, and internal MOS memory
- The front-end diagnostic and console PDP-11 subsystem that also supports UNIBUS unit-record equipment
- The I/O subsystem that consists of integrated channels/controllers and PDP-11-based communications subsystems

Figure 5-1 illustrates a fully configured KL10-E central processing unit. Key elements of this implementation of the processor are internal memory, a MASSBUS for high-speed disks and tapes, and a UNIBUS for low-speed peripherals (unit record and PDP-11-based communications interfaces). The MASSBUS terminates in RH20 integrated channels/controllers, and the UNIBUS terminates in the DTE interface.

Figure 5-2 illustrates the 60-inch-high DIGITAL Corporate "High-Boy" cabinets in which the KL10-E is packaged.

EXECUTION BOX (EBOX)

The execution box (EBox) is the part of the processor that actually executes instructions and performs various control functions for overall CPU operation. The KL10-E has 398 microprogrammed instructions. Each instruction is implemented as a series of microinstructions that performs various logical functions such as processor state control, data path control, and the actual implementation of each instruction. Using microcode as opposed to "hardwired" instruction implementation, a machine instruction executes one or more microstore instructions. To execute a given machine instruction, the microinstruction sequence associated with that instruction performs functions such as the data movement, operations on data, or control steps needed to execute that instruction. The microprogrammed KL10-E central processor offers several important features:

- The CPU executes TOPS-20 paging (memory management) and monitor call interface
- Engineering changes or central processor improvements can often be implemented by microcode changes, rather than by wiring changes, since the microcode is loaded by the PDP-11 front-end processor during system start-up
- Cache and virtual memory operations are enhanced
- The packaging technology of microcoded hardware permits smaller size and greater flexibility in instructions

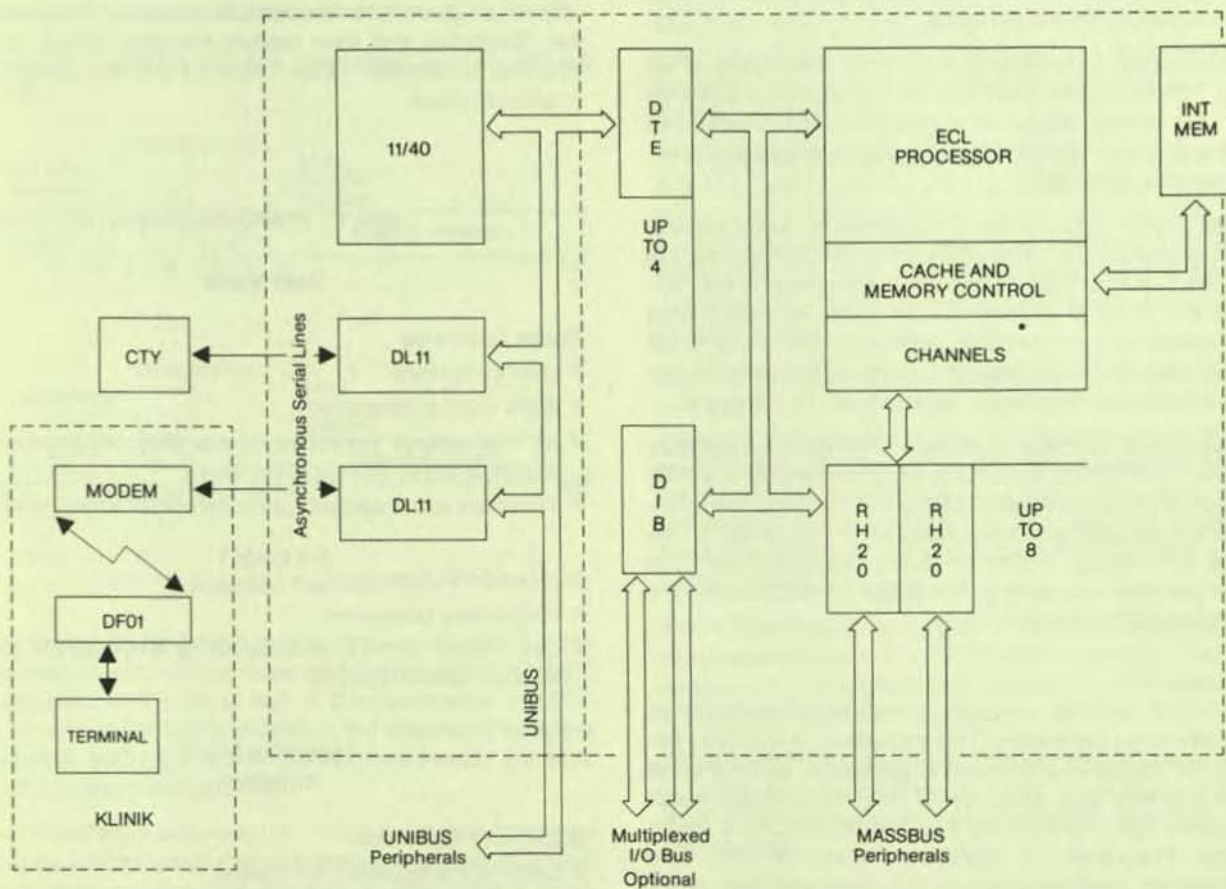


Figure 5-1
KL10-E Central Processing Unit

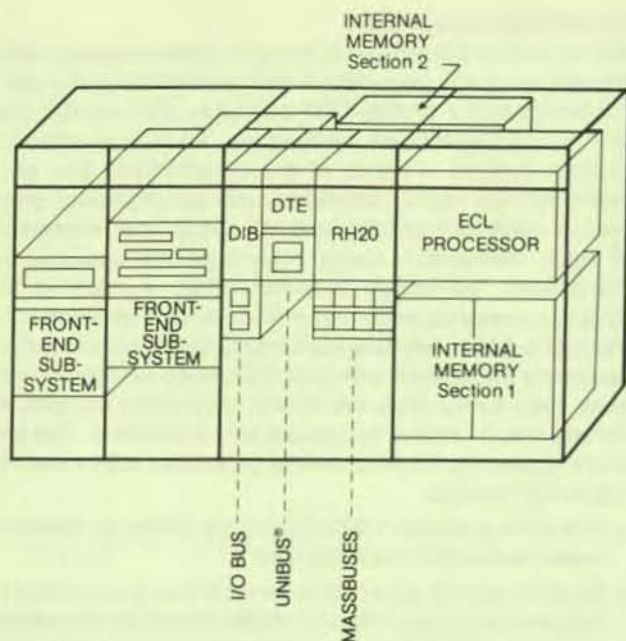


Figure 5-2
KL10-E Mechanical Configuration

- Architectural extensions can often be made using microprogramming, obviating the need for extensive engineering change orders (as, for example, in the support of new peripheral subsystems)

The microcode is actually implemented with 2,048 75-bit words. The EBox also has a 512-word instruction dispatch table that rapidly decodes instructions and dispatches operations to the appropriate microstore sequence to implement the instruction.

Besides performing the basic functions of decoding and implementing control store steps for the implementation of each instruction in the KL10-E set, the microcontroller executes the more fundamental operations of sequencing the program and processing interrupts. The KL10-E set also handles TOPS-20 paging operations beyond the basic address translation made, for example, by the pager.

The EBox also contains eight sets of 16 general purpose registers, four timing and accounting meters, and a DTE interface to the low-speed UNIBUS I/O devices. The interface to the low-speed control devices is controlled by the EBox's microcode. It performs the required data and control transfers by stealing microcode cycles between the execution of CPU instructions.

Instruction Set

The KL10-E has 398 microprogrammed instructions, an extremely large repertoire. This provides the flexibility required for specialized computing problems. Since the set provides a very wide selection of instructions from which to choose, few are typically needed to perform a given function. Programs can therefore be shorter than they would be on other computers. The large instruction set also simplifies the monitor (operating system), language processors, and utility programs.

In addition to the 398 instructions, the KL10-E provides 64 programmable operators, of which 33 trap to the monitor and 31 trap to the user's call area. The remaining instruction codes are unimplemented and reserved for future expansion. An attempt to execute one of the unimplemented instructions results in a trap to the monitor.

Despite its size, the instruction set is easy to learn. It is logically grouped into families of instructions, and the mnemonic codes are constructed modularly. All instructions are capable of directly addressing 256K 36-bit words of memory without resorting to base registers, displacement addressing, or indirect addressing. Instructions can, however, use indirect addressing with indexing to any level. Most instruction classes, including floating point, allow immediate mode data, the result of which is effective address calculation used directly as an operand to save storage and speed execution.

The *processor mode* concept has been a keystone in DIGITAL's timesharing or multitasking systems for more than a decade, and it is central to the KL10-E implementation. Instructions are executed in either *user mode* or *executive mode*. In a multitask environment a user must not execute instructions that would jeopardize other users. With the KL10-E, attempted execution of such an instruction in user mode will trap to the operating system analyzing the instruction, and appropriate action will be taken.

In executive mode operation any implemented instruction is legal. The monitor operating in executive mode is able to control all system resources and the state of the processor. Executive and user modes are each further divided into two submodes. Table 1 defines the processor mode implementation.

Table 1 Processor Modes

User Mode

Public Submode

- User programs
- 256K word address
- All instructions permitted unless they compromise integrity of the system or other users
- Transfers to concealed submode only at entry points

Concealed Submode

- Proprietary programs
- Can READ, WRITE, EXECUTE, or TRANSFER to any location labelled public

Executive Mode (monitor)

Supervisor Submode

- General management of system
- Those functions that affect only one user at a time
- Executes in virtual address space labeled public

Kernel Submode

- I/O for system
- Those functions that affect all users

Instruction Format — In all but input/output instructions, 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 always supplies information for calculating the effective address that is used for immediate mode data, or that is the actual address used to fetch the operand or alter program flow. The effective address is referred to throughout this document as the user (or operating system) virtual address. Memory mapping and/or paging hardware can map this effective address into a very different physical address within main memory. However, the programmer need only be concerned with an effective user address space that can be 256K words. 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 a memory address that is used as the basis for effective address calculation or as an actual operand in immediate mode format.

All input/output instructions are designated by "ones" in bits 0–2. Bits 3–9 address the device to be used, and bits 10–12 specify the operation. Bits 13–35 are the same as for non-I/O instructions. They are used in calculating an effective address for the data to be transferred.

Figure 5-3 illustrates the two instruction formats for the KL10-E.

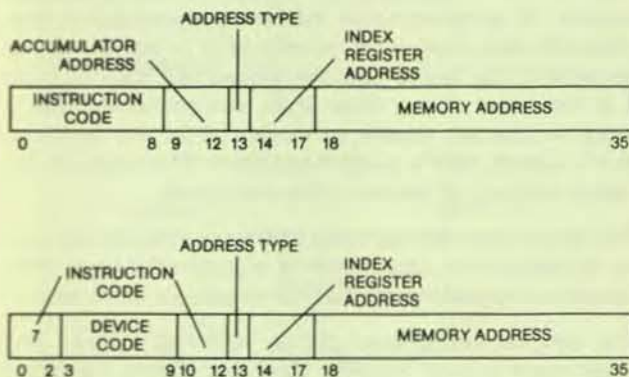


Figure 5-3
Instruction Formats

Half-Word Data Transmission — Half-word data transmission instructions move half a word and can modify the contents of the other half of the destination location. The 16 sets of instructions differ in the direction they move the chosen half-word and in the way they modify the other half of the destination location.

Full-Word Data Transmission — Full-word data transmission instructions move one or more full words of data from one place to another. The instructions can perform minor arithmetic operations such as calculating the negative or the magnitude of the word being processed.

Byte Manipulation — Five byte-manipulation instructions pack or unpack bytes of any length anywhere within a word.

Logic Instructions — Logic instructions shift, rotate, and execute the 16 Boolean operations on two variables.

Fixed-Point Arithmetic — The KL10-E is a 2's complement machine in which zero is represented by a word containing all zeros. As in comparable implementations, the KL10-E logic does not keep track of a binary point. The programmer must adopt a point convention and shift the magnitude of the result to conform to the convention used. Two common conventions are to regard a number as an integer (binary point to the right) or as a proper fraction (binary point to the left). In these cases, the range of numbers represented by a single KL10-E word is -2^{35} to $2^{35}-1$ (binary point to the right) or -1 to $1-2^{-35}$ (binary point to the left). Since multiplication and division use double-length numbers, there are special instructions with integral operands for these operations.

The format for double-length, fixed-point numbers is an extension of the single-length format. The magnitude (or its 2'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; bit 0 of the low-order word is ignored. The range for double-length integer and proper fractions is thus -2^{70} to $2^{70}-1$ (binary point assumed at the right) or -1 to $1-2^{-70}$ (binary point assumed at the left). Zero is represented by a word containing all zeros.

Floating-Point Arithmetic — The KL10-E processor has instructions for scaling, negating, adding, subtracting, multiplying, and dividing in single- and double-precision floating-point format.

In single-precision floating-point format, one bit is used for the sign, eight bits for the exponent, and 27 bits for the fraction. In double-precision floating-point format, one bit is used for the sign, eight bits for the exponent, and 62 bits for the fraction. This results in a precision in the fraction of 1 part in 4.6×10^{18} and an exponent of 2 in the range from -128 to $+127$.

Normalized single-precision floating-point numbers have a fraction that can range in magnitude from $\frac{1}{2}$ to $1-(2^{-27})$. Increasing the length of a number to two words does not significantly change the range, but it does increase the precision. In any format the magnitude range of the normalized fraction is from $\frac{1}{2}$ to 1 less the value of the least significant bit. In all formats the exponent range is from 2^{-128} to 2^{127} .

Fixed/Floating Conversion — Special instructions convert between fixed- and floating-point formats. Two sets of conversion instructions are provided, one optimized for FORTRAN, the other for ALGOL.

Arithmetic Testing — An arithmetic testing instruction can jump or skip depending upon the result of an arithmetic test. It can also perform an arithmetic operation on the tested word.

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

Program Control — Program control instructions include several types of jump instructions and subroutine calls including calls that save the return address on a pushdown stack.

Input/Output Operations — Input/output (I/O) instructions govern all direct transfers of status to and from the peripheral equipment. They also perform many operations within the processor. Block transfer instructions handle bulk data transfers to and from medium-speed I/O bus devices. High-speed devices transfer data directly to memory through internal channels.

Unimplemented User Operations (UOUs) — Many of the codes not assigned as specific instructions are executed as unimplemented user operations. The word given as an instruction is trapped and must be interpreted by a routine included for this purpose, either by the programmer or by the monitor.

TOPS-20 uses the instruction JSYS to transfer control from the user program to the monitor where any of a large number of functions is performed. The function performed is determined both by the effective address and by arguments in the user's registers.

Business Instruction Set — There are five classes of instructions in the Business Instruction Set of the KL10-E central processor. Four of these are arithmetic instructions to add, subtract, multiply, and divide using double-precision, fixed-point operands. The STRING instruction in the fifth class can perform nine separate string functions.

These functions include 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, and compare-string in both offset and translated mode. Offset mode is byte modification by addition of the effective address of the string instruction. Besides providing the translation function, these instructions can control AC flags and can detect special characters in the source string.

The Business Instruction Set provides faster processing because there are special instructions for a variety of string operations. These instructions can be used on many code types such as ASCII and EBCDIC. The Business Instruction Set exemplifies a microprogrammed processor's advantages in meeting special user needs.

Trap Handling — The execution of programmed trap instructions permits the KL10-E to directly handle arithmetic overflows and underflows and pushdown list overflows.

Fast Register Blocks — The KL10-E architecture includes eight sets of 16 general purpose registers. They can be used as accumulators, index registers, or as the first 16 locations in main memory. Since register addressing is included in the basic instruction format, no special instructions are needed to access these registers. Different register blocks are used for the operating system and individual users, eliminating the need for storing register contents when switching from user mode to executive mode.

Programmable Address Break — When a specified location is properly referenced, a programmable address

break suspends a user program and traps to the operating system. This facility is particularly useful in program development and debugging.

Meters

Two meters built into the KL10-E provide a number of timing and counting functions, including an interval timer and a time base counter.

The meters are controlled by I/O instructions to internal devices. Many of the timing functions utilize a microsecond pulse source originating in the basic 30-megacycle machine clock. Designed with a tolerance of 0.005 percent, this pulse source drifts less than five seconds over 24 hours.

The *interval timer* is a programmable source of interrupts with a one-microsecond resolution. Used for realtime applications and for page management by the monitor, it can implement a realtime deadline schedule with varying deadlines.

The *time base* is a 60-bit, one-microsecond resolution counter that can generate a source of elapsed time. This 60-bit register provides more than 9,000 years maximum time before wrap-around.

Priority Interrupt System

The KL10-E has a powerful, flexible priority interrupt system that permits overlapped concurrent asynchronous operation of the central processor, peripheral controllers and devices, and software service routines.

Devices are assigned under program control to any one of seven priority levels through dynamic loading of a 3-bit register within the device. Each of the seven levels has any number of programmable sublevels with which a programmer can change the priority level of any device, or disconnect the device from the system and later reinstate it at the same or any other level. In a similar manner, a program can set, enable, or disable all (or any combination of) levels with a single instruction. The program can assign some or all devices to the same level.

The system can also generate interrupts through software, so hardware can operate on a high-priority level while related computations can be performed on a lower level.

The program-assignable priority interrupt system provides much greater flexibility than permanently hardwired systems because hardwired systems require a large number of levels, often operate with an extremely high overhead, and cannot change device priorities without system shutdown and rewiring.

In conjunction with the priority interrupt system, a set of instructions (BLOCK-IN/BLOCK-OUT) allows the transfer of blocks of information between a device and memory in a single operation. When an interrupt occurs, the KL10-E will trap to a predetermined location. The BLOCK-IN/BLOCK-OUT instruction identifies the source of the interrupt, transfers a word in or out, updates word count and data address, and dismisses the interrupt. When the word count is zero, a different action occurs that allows the program to either process the block of information that has been transferred or to move on to other activities.

MEMORY SUBSYSTEM

Central to the KL10-E architecture is the implementation of all memory control through a discrete memory controller (MBox). The MBox is responsible for all memory requests from the EBox and integrated channels/controllers for high-speed peripherals (disks and tapes). The MBox does paging (address translation) for memory management and contains the cache memory for CPU models so equipped.

Internal Memory

The internal memory implementation of the KL10-E offers improved reliability and lower cost per word than external memories, due to its simplified design and, since no long bus interfaces are required, fewer components.

Table 5-2 Internal Memory System

Feature	MF20 Memory System
	MF20 (MOS)
Word Size	36 bits plus 7 error-correcting bits
Minimum Memory Size*	256K words
Maximum Memory Size*	2,048K words
Read Access Time**	467 ns
Read Cycle Time**	
1-word	667 ns
4-word	1,267 ns
Interleaving	4-way
Minimum memory unit that can be disabled	64K words
* Within CPU cabinet	
** Measured at memory	

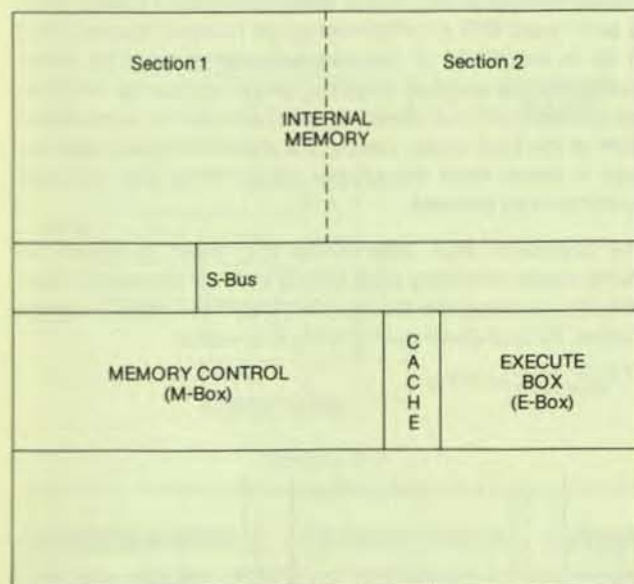


Figure 5-4
Internal Memory Systems Configuration

MOS Memory

Metal Oxide Semiconductor (MOS) memory has a word length of 44 bits (36 data bits, seven error detection and correction bits, and one spare bit). This implementation allows hardware detection and correction of a single-bit data error and hardware detection of a double-bit error. MOS memory is available in 256K word increments to a maximum of 2M words.

If a correctable (single-bit) error is detected, the hardware automatically corrects the data word, and notifies the CPU with a flag. This allows the program to continue without consideration of such memory deterioration while, at the same time, the operating system can gather statistics concerning possible failing areas of memory. If a detectable but noncorrectable (double-bit) error is encountered, the memory controller signals the CPU, sets a noncorrectable-error flag, and provides certain other data on the first error detected. The operating system will mark the page containing this location and will not reuse it.

The spare bit can be substituted (upon software command) for failing bits, significantly enhancing the mean time between failure (MTBF) for this type memory.

Cache Memory

Cache memory decreases instruction execution time by substantially speeding the average memory reference. This is accomplished by placing a high-speed semiconductor memory (the cache) inside the MBox. The cache can hold up to 2K words from memory. Whenever the program accesses a word held in cache, the request is satisfied in 133 nanoseconds, compared to 867 nanoseconds or more for a memory reference as measured at the EBox.

The success of cache memory depends on the quality of the algorithm used to decide which 2K words from memory are cached. Cached locations change constantly with varying system demands, but the algorithm is based on the assumption that memory references tend to be somewhat localized. In a typical program the flow of control is linear within a narrow scope. If an instruction has just been executed from location N, there is a high probability that there will soon be an instruction executed from location N + 1.

The "hit rate" of the KL10-E cache memory usually exceeds 90 percent.

Several unique design features of the KL10-E cache memory are:

- KL10-E cache is not a write-through cache. If the EBox instructs the MBox to write a given location, the location is modified only in cache. The corresponding physical location will be updated only when the monitor instructs the MBox to sweep cache or when a quadword (four adjacent 36-bit words) must be emptied to make room for new data.
- The cache is organized to handle physical addresses. Cache as used on some other large systems, however, is oriented toward virtual addresses. Stanford University developed the cache algorithm used in the KL10-E, and its modeling demonstrates that using physical addresses is more efficient than using virtual addresses.

- The hardware's use of the cache is dependent upon the M Box microcode that is normally set up to support all four cache pages. However, if desired, some or all of the cache can be turned off. This option is exercised when the front end is initializing the KL10-E at system startup.

Organization

The cache can hold up to 2,048 words from memory. It is organized as four separate, 512-word pages that operate in parallel.

Each cache page has a directory associated with it. The directory consists of 128 13-bit entries. A single directory entry contains information concerning four words of data within the cache page (Figure 5-6).

A four-word cell described by a single directory entry is called a quadword. The 13-bit directory entry for a quadword contains the physical page number of the page in memory that originated the quadword. In turn, the position of a word within a cache page is always the same as the position of the word in its original page of memory.

In the following simplified example, assume that there is only one page of cache and its associated directory, rather than the four that are actually provided in the hardware. Suppose that the EBox has requested the contents of the 22-bit physical address 14707002 (addresses are in octal notation). The MBox must determine if it must read memory location 14707002, or if that location is already in cache. The first step is to split the physical address into a 13-bit physical page number, in this case 14707, and a 9-bit index into the page 002. In other words, the search is concerned with the 002nd word of physical page 14707. If this word is already cached, the only place it could be in a single cache page would be word 002, because the position of a word within a cache page is always the same as the position of the word in its original page. The MBox must examine the 13-bit directory entry corresponding to the 002nd word of the page and compare it to the desired physical page number of 14707. If the directory entry holds 14707, then the 002nd cache page word is indeed the required word. If the comparison fails, then it must read physical memory.

Since there is exactly one directory entry for each quadword, it follows that all four words in the quadword must come from the same physical page. Moreover, a word must have the same position in the cache page that it had in the physical memory page. These facts imply that the four words in a single quadword are physically contiguous in memory as well as in cache — a key concept in the KL10-E cache design.

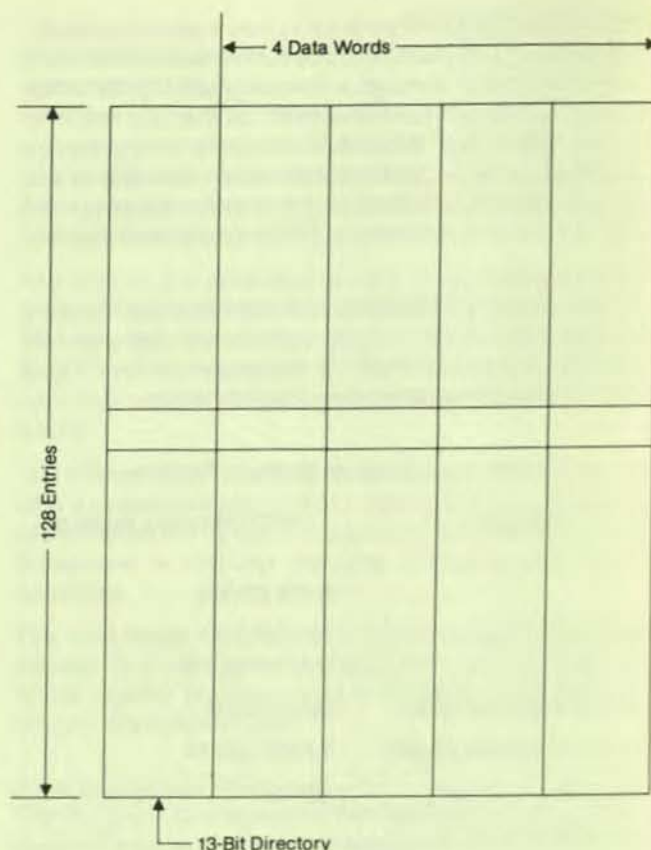


Figure 5-5
Cache Page Structure

The above example was simplified by the omission of three-fourths of the cache pages. In a real system with four pages (2K words) of cache, a given physical word might actually reside in any one of the four pages of cache. It has to be in word 002 of whichever page holds it; just as it had to be in word 002 of the single cache page. The MBox compares the desired physical page number of 14707 to the contents of four directory entries, one for word 002 in each of the four cache pages. If a match is found, then the data is taken from the proper page. Otherwise physical memory must be read.

The algorithm that determines the order in which the cache pages are filled uses tables in RAM storage. Therefore, by rewriting the tables it is possible, under program control, to shut down part or all of the cache.

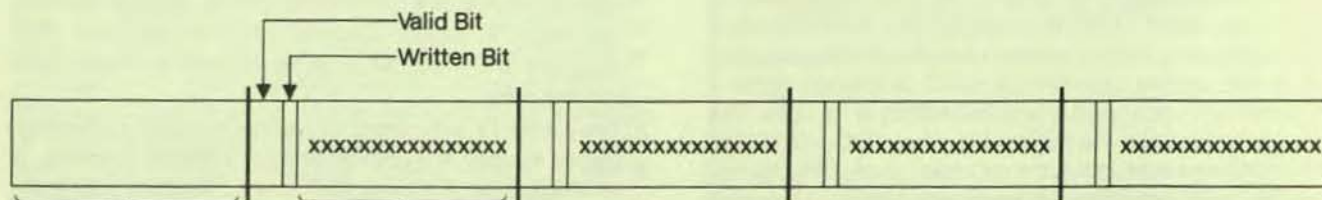


Figure 5-6
Single Entry in Cache

System Control of Cache

There are three different operations to which the monitor can subject the cache: invalidation, validation, and unloading. These operations can be performed on the entire cache or on entries belonging to a single memory page.

To invalidate a location means to clear its valid and written bits, thus emptying the location. Validation of a location means that, if an entry has been written since it was brought in from memory, then the modified contents must be written back into physical memory. This is necessary because the cache is not a write-through cache. The unloading of a location requires the MBox to validate the location and then to invalidate it.

Memory Mapping on the KL10-E

The KL10-E provides a hardware implementation of memory address mapping from a program's memory address space (effective address or virtual address) to the physical memory address space. During mapping, the most significant bits of the memory address are changed dynamically to indicate the physical address at which the program is running. Mapping provides access to the entire physical memory space, which can be up to 16 times larger than the maximum user address space. The user's effective address space is 256K words addressed with 18-bit addresses. The physical address space is 4M words addressed with 22-bit addresses (where 4M is equivalent to 4,194,304 decimal). Provisions exist in the KL10-E to allow a user address space larger than 256K words.

The memory mapping process uses the nine most significant bits of the virtual address as an index into the appropriate user or executive page map. The data located by the index provides 13 bits that are appended to the nine least significant bits of the virtual address to form the 22-bit physical address. Also provided are three bits that indicate what type of memory requests are allowed to the page in question, for example, none, read-only, or proprietary. Figure 5-7 illustrates this mapping by means of the page table.

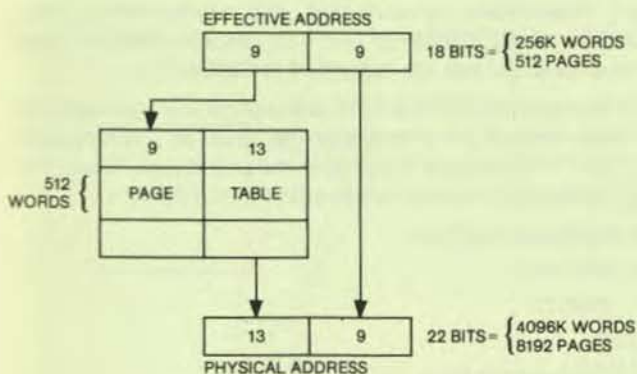


Figure 5-7
Memory Mapping

If the address is in the range 0 - 17₈, inclusive, the hardware fast register blocks are referenced instead of the main memory system. The user mode bit and the nine high-order bits of the virtual address are used to perform a "table lookup" in the hardware page table. If a page table

entry exists, the contents of the entry supply the 13-bit most significant part of the physical memory address and also supply three bits that indicate types of memory references allowed to this page. If the memory request is consistent with the request type allowed, the physical address used consists of the 13 bits from the related page table entry as the most significant bits of the physical address, and the nine least significant bits of the effective address as the least significant bits of the physical address.

When the relocation data for a referenced page does not exist in the hardware page table, the microcode reads the relocation data from the page table in memory, stores it in the hardware page table of the KL10-E, and proceeds.

The operating system assigns the memory area for each user by loading the various in-memory page tables, setting up the trap locations in the user page map, and responding appropriately when a trap occurs. The monitor provides memory protection for itself and for each user by filling the page tables only with entries that are allowed to be accessed. This makes it impossible for a user to infringe on another user's physical memory area.

The major benefits of this paging capability are:

- Full memory protection of one user from another through hardware and microcode
- Freedom to scatter the pages of a user virtual memory area throughout physical memory (programs do not have to be physically contiguous), eliminating memory shuffling and complex memory management even though a user program grows during execution
- The opportunity to execute a program when all of its pages are not in physical memory (i.e., a virtual memory capability)

FRONT-END SUBSYSTEM

A key element in the operation and maintenance of the KL10-E is the PDP-11-based console/diagnostic front-end computer. This minicomputer provides all console functions for the KL10-E and its associated operating systems (TOPS-10 and TOPS-20).

The PDP-11 interfaces to the KL10-E through a dedicated DIGITAL Ten-to-Eleven (DTE-20) interface with a full-capability UNIBUS. The DTE-20, under control of both the KL10-E and the front-end PDP-11, can examine or deposit words into memory, and can provide two-way data transfers. Asynchronous communications lines for the console terminal and a dedicated line for remote/local diagnostic functions (KLINIK) connect the user to the operating system through the console front end.

In addition to the DTE-20 link, both the KL10-E (through an RH20) and the PDP-11 (through an RH11) are interfaced to separate ports on an RP06 disk drive. Figure 5-8 illustrates these connections.

In addition to the console and diagnostic functions, the KL10-E's front-end console/diagnostic PDP-11 supports unit record and asynchronous communications equipment. From the user standpoint, these devices operate identically to those interfaced to separate PDP-11s. They are explained in detail in the section below on I/O devices.

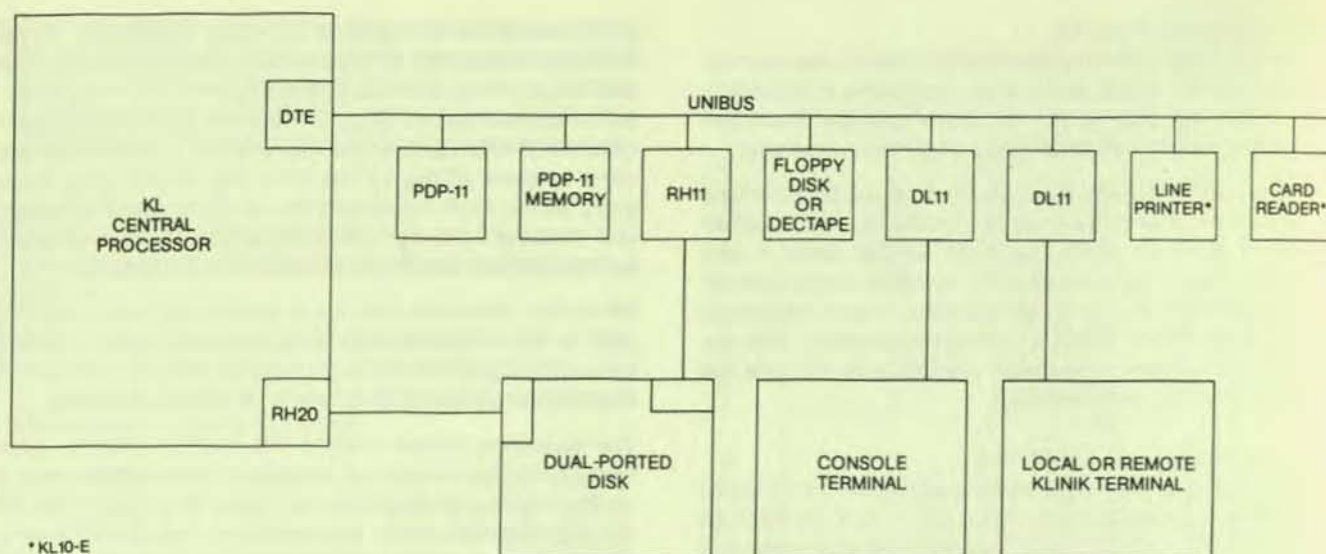


Figure 5-8
Front-End Subsystem

All KL10-based systems rely on the front-end for at least two basic functions. First, the PDP-11 is used to initiate CPU operations from a dead stop. This process involves setting up KL10-E status, loading microcode, configuring memory, and starting the monitor bootstrap. These operations are conducted primarily across a diagnostic bus that connects the front-end to both the EBox and the MBox by software. Second, the PDP-11 supports the console terminal with which an operator can control the system. It is this terminal that governs the operating system and the tasks running under it. The terminal can also be used as a normal user terminal.

These functions are implemented using a special operating system that runs in the front-end with supporting code for TOPS-20.

The PDP-11-based operating system is built around DIGITAL's RSX realtime system software. RSX allows concurrent operation of multiple tasks. One task is the "command parser," the program that recognizes commands typed on the console terminal and passes appropriate messages through the DTE-20 to the TOPS-20 operating system. Other tasks include KLINIT, which oversees initialization of the KL10-E processor, and KLINIK, which provides a diagnostic environment for diagnosis and control of the KL10-E by privileged users from either local or remote locations.

Some devices associated with this front end support console and diagnostic operations and can be used as peripheral devices. These include the RH11 disk controller and a floppy-disk drive. The floppy-disk drive can be used as an alternate bootstrap device if, for some reason, the RP06 disk either cannot be used or contains incorrect information.

Front-end operations require that the dual-ported RP06 be available for normal operations. The RP06 is connected to both the KL10-E and the PDP-11 through dual-ported hardware. The disk used is KL10-formatted with a section

formatted specifically for the front-end files. There are both hardware and software interlocks to prevent the KL10-E and PDP-11 from interfering with one another. The PDP-11 disk area is not available for user storage.

KL10-E systems can support up to four PDP-11s; however, only one of these can be the controlling front end. To prevent conflicts between different PDP-11s, the operations described in this section can be accomplished only through a "privileged" hardware- and software-selectable DTE-20.

INPUT/OUTPUT SUBSYSTEM

The KL10-E input/output subsystem provides standard interfaces and control logic for connecting the KL10-E to a wide range of peripheral and communications equipment. This includes disks and magnetic tapes, and — through PDP-11-based communications subsystems — cardreaders, lineprinters, synchronous, and asynchronous lines. Details of the operation and the specifications of these peripheral devices are described in Section 7.

To standardize the interface and permit the connection of a wide range of peripheral devices, DIGITAL has long used a "bus" architecture. Three external buses (see Figure 5-9) are available to communicate with the KL10-E CPU:

- Multiplexed I/O bus
- MASSBUS
- UNIBUS

Multiplexed I/O Bus

A multiplexed I/O bus is optional on the KL10-E. TOPS-20 supports unit record and communications capability through PDP-11-based controllers but does support a few special devices on the I/O bus. See Section 7 for information about these peripheral devices. The I/O bus provides control and a character-by-character, interrupt-driven interface for low-speed devices. Figure 5-10 illustrates the hardware associated with the KL10-E I/O bus.

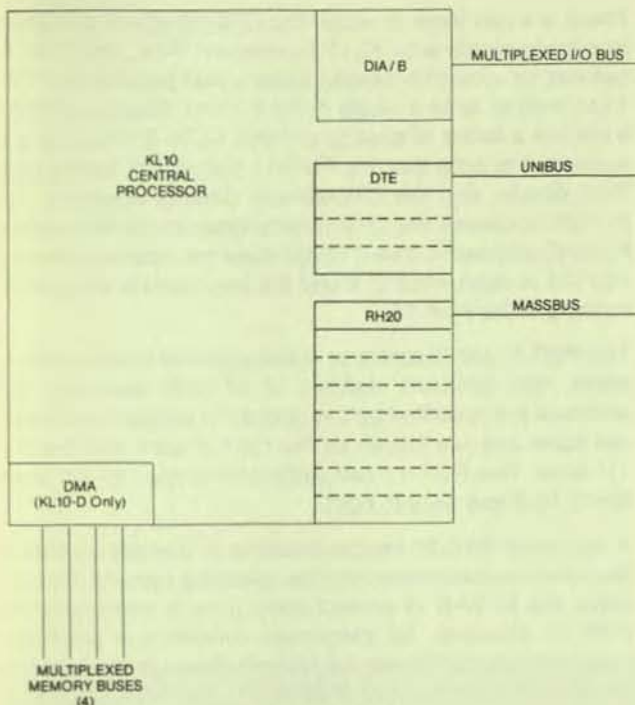


Figure 5-9
KL10-E Input/Output Bus Architecture

MASSBUS

DIGITAL's MASSBUS allows connection of a wide range of DIGITAL disks and magnetic tapes to the KL10-E central processor. See Section 7 for information about MASSBUS peripheral devices. Each MASSBUS can interface as many as eight peripheral device controllers to a single RH20 MASSBUS controller. Each RH20 interfaces to the EBox through the EBus and to one of eight data channels located in the MBox through the channel bus (CBus). (See Figure 5-11.)

The channel also includes a series of multiple-word channel buffers to support high-speed I/O between the KL10-E

memory controller and individual MASSBUS devices. Up to eight RH20s can be installed in the KL10. For each device, the channel provides a 15-word data buffer, a channel command word register, and a control command word pointer that serves as a program counter. The channels transfer data independently of the KL10-E execution logic by executing channel programs placed in memory by device service routines that use memory cycles. The device service routines are in the operating system.

A channel bus interfaces the RH20 integrated controllers to the memory control unit (MBox) of the KL10-E CPU. The bus is a physically short, high-speed data transfer path designed for peak I/O bandwidth of six million words per second — well in excess of that encountered in any existing memory or peripheral units — between the memory control unit and the MASSBUS controllers (RH20s). Operating in synchronous time-division multiplexed mode, the channel bus allows the connection of multiple RH20s in memory.

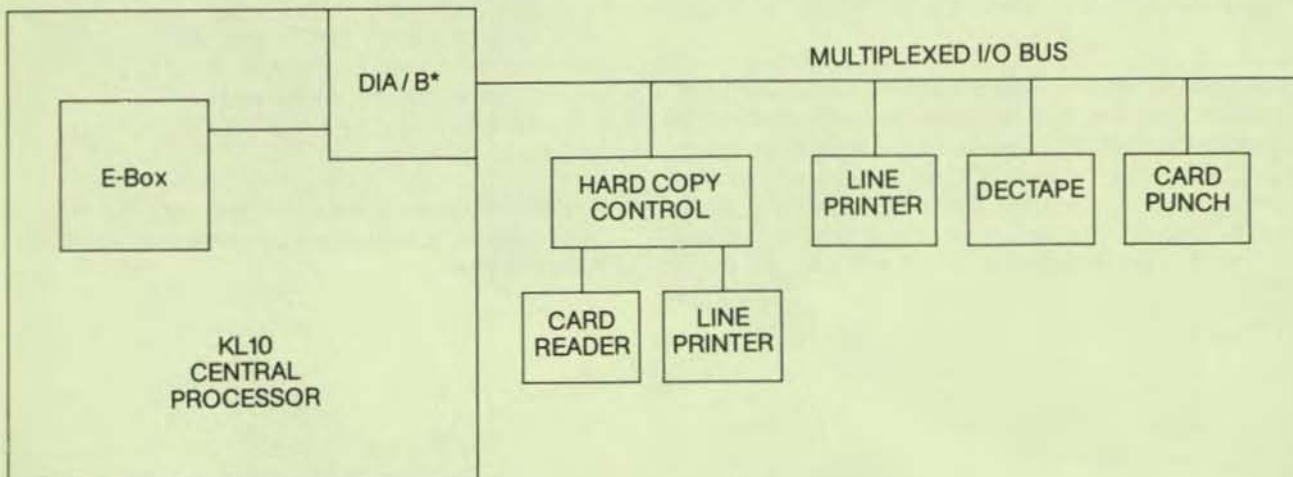
Each RH20 terminates in a MASSBUS. This high-speed data path connects the RH20 integrated controllers and mass storage devices such as disk or tape. Operating either asynchronously or synchronously, the MASSBUS transfers control and status information or blocks of data between devices and controllers.

DIGITAL disks and tapes connect directly to the MASSBUS, providing an integrated, high-speed subsystem for mass storage.

UNIBUS

DIGITAL's UNIBUS provides the third path between external devices and the KL10-E CPU. As illustrated in Figure 5-12, this interface uses the DTE-20 interface to provide:

- The ability to examine or deposit words in PDP-11 memory and specified areas of KL10-E memory (under the control of a PDP-11 processor connected to the DTE-20-terminated UNIBUS or the KL10-E CPU)
- High-speed simultaneous transfer of data between PDP-11 and KL10-E memory. This transfer operates in a



*DIA-standard on KL10-D.

Figure 5-10
Multiplexed I/O Bus Architecture

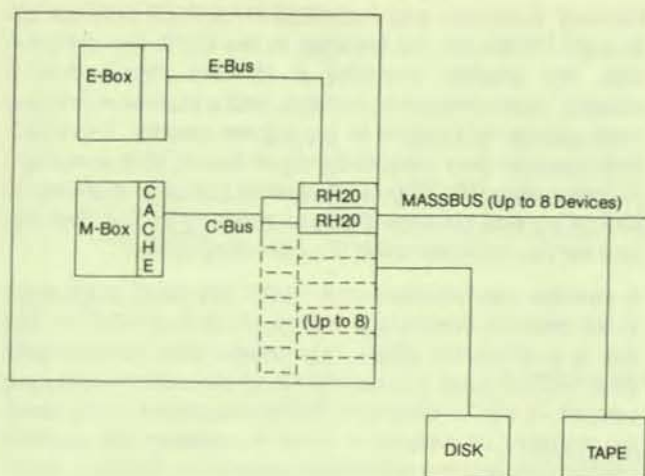


Figure 5-11
MASSBUS Interface

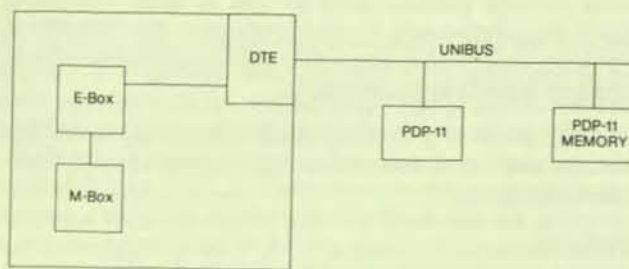


Figure 5-12
UNIBUS Interface

PDP-11 "byte" mode and transfers 8- or 16-bit bytes under the control of either the PDP-11 or KL10-E processors.

- Doorbell interrupts — the PDP-11 can cause an interrupt to the KL10-E processor and vice versa

The DTE-20's two operating modes, restricted and privileged, are controlled by a hardware switch on the DTE-20. The privileged DTE-20 is used only for the console/front-end PDP-11; it is discussed in the section on Front-End Subsystems in this document. Only the restricted DTE-20, used to control peripheral devices and communications equipment, is described here.

There are two ways in which the DTE-20 allows a PDP-11 to communicate with KL10-E memory. First, the PDP-11 can use an examine/deposit feature that permits the PDP-11 to read or write a single KL10-E word. Second, DTE-20 transfers a string of data to or from KL10-E memory. It is important to note that the PDP-11 memory is itself a UNIBUS device, and the DTE-20 can directly access it. For KL10-E accesses, the DTE-20 interfaces to the EBox of the KL10-E processor. The DTE-20 does not operate unless a PDP-11 is connected to it and the appropriate program is running in the PDP-11.

The PDP-11 can examine or make deposits to any address within two windows defined in KL10-E memory. The windows are specified by the operating system exec process table and are known as the "to-10" area and the "to-11" area. The PDP-11 can write only to the "to-10" area; the KL10-E can write to both.

A restricted DTE-20 cannot examine or deposit outside of the windows established by the operating system. This enables the KL10-E to protect itself from a malfunctioning PDP-11. However, for diagnostic functions, a privileged front end used only as the console/front-end processor can examine and deposit anywhere in KL10-E memory.

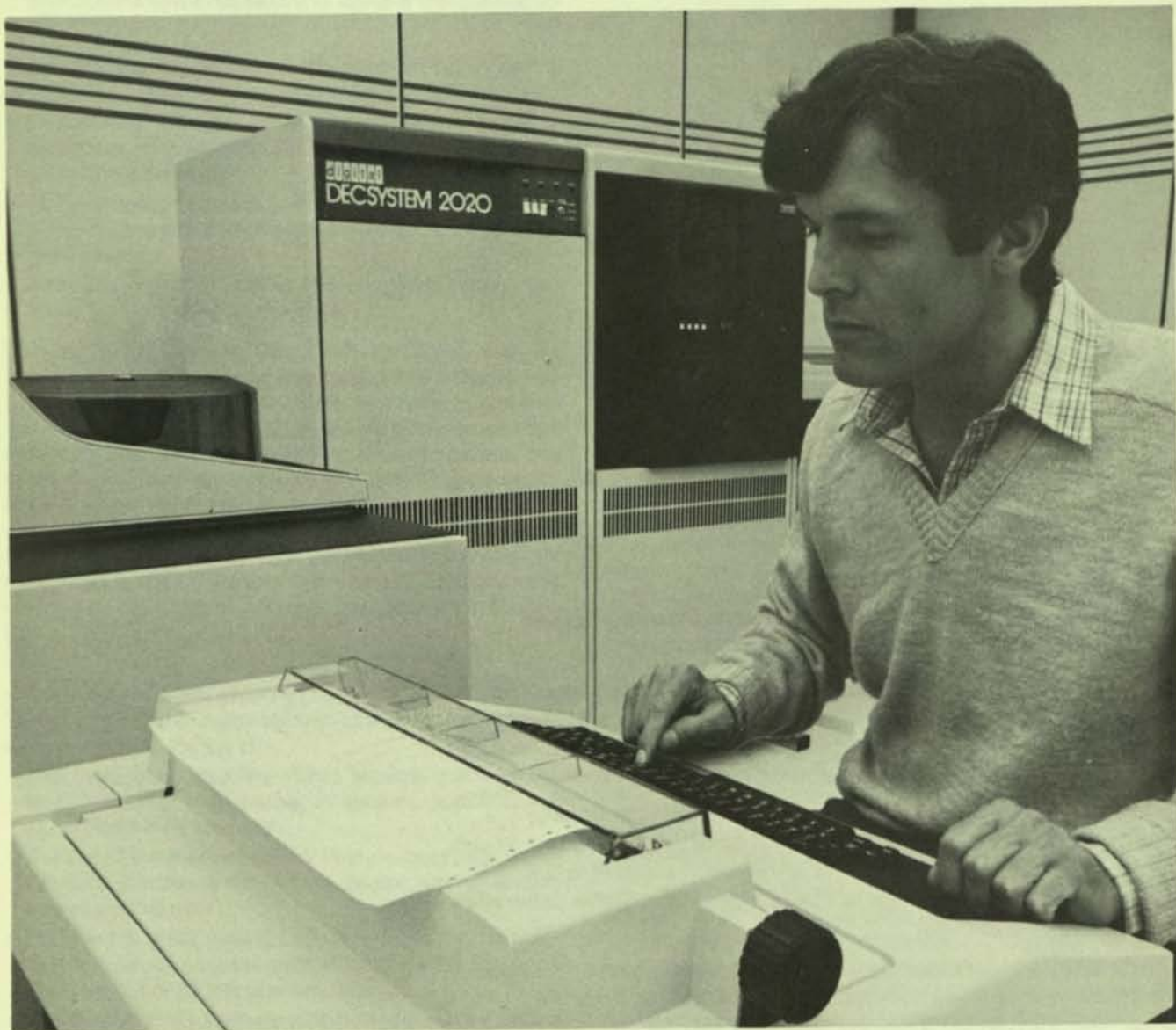
The other transfer mechanism, byte transfer, permits transfers to or from anywhere in KL10-E memory. Byte size can be eight or 16 bits at the program's option, and byte transfer supports concurrent "to-10" and "to-11" transfers.

Once a transfer has been initiated, the DTE-20 handles it without further intervention from either CPU at the program level. The operating system will not be interrupted until the entire transfer is complete. The DTE-20 recognizes the end of the transfer by the expiration of a byte counter. Transfer completion results in an interrupt on the assigned PI level. On the KL10-E side, actual implementation of a byte or word transfer is by microcode. Thus, during transfers through the DTE-20, cycles stolen from the microcode incur some CPU overhead.

As indicated earlier, DTE-20 operation requires a PDP-11 processor. While service routines are integral to the operating system, it is the PDP-11 program that actually controls the peripheral equipment. Cardreaders and lineprinters are supported on the console/front-end processor, as are a limited number of asynchronous communications lines.

You can obtain additional information on any peripheral or communications subsystem by contacting your DIGITAL representative.

6 KS10 Processor



The KS10 central processing unit (CPU) forms the basis of the DECSYSTEM-2020.

The KS10 has changed mainframe computing. People constrained by budgets to settle for something less than a mainframe can now install a KS10-based DECSYSTEM-2020 with full assurance that software-compatible growth is easy, cost-effective, and straightforward.

The DECSYSTEM-2020 comprises a number of major units or subsystems that communicate with each other. The minimum system has five subsystems: processor, MOS memory, console, a UNIBUS adapter to handle the disk system, and a UNIBUS adapter to handle all other peripheral devices.

The console, which is based on a microprocessor, boots the system from disk and handles interaction of the operator or a remote diagnostic link with other subsystems.

SYSTEM ARCHITECTURE

The KS10 microprogrammed central processor forms the basis for the DECSYSTEM-2020 computer system. The DECSYSTEM-2020 consists of four major subsystems:

- KS10 CPU and memory
- Console and diagnostic microprocessor
- UNIBUS Adapters
- Peripheral controllers

KS10 Technology

The KS10 uses low-power Schottky TTL and features a four-bit data path slice microprocessor to provide full TOPS-20 functionality at low cost.

Microprogramming technology used in the KS10 design relies heavily on experience gained in the design and implementation of the successful KL10 central processor and several of DIGITAL's smaller central processors. Microcoded machine instructions are implemented by one or more microstore instructions that move or operate on data or that control steps necessary to achieve the desired effect.

Microprogramming implementation of the KS10 central processor provides several important advantages:

- Engineering changes or central processor improvements can often be microstore (firmware) changes rather than wiring changes
- The packaging technology of microcoded hardware reduces machine size and increases flexibility of machine instructions
- Microprogramming speeds virtual address cache and virtual memory control operations

The microprogramming instruction combines with the four-bit-slice technology to make the KS10 reliable, low-cost, flexible, and compact. Chip technology has progressed to the point where a single chip contains an entire data path, including data registers, data operations, and testing and control functions for four bits. Thus, to implement a 36-bit KS10 central processor, nine four-bit data path slice chips are used, plus a tenth four-bit data path slice chip for extra control bits. Control lines from each are connected to the microstore logic, so the microcode can dictate function, control, and data flow throughout the system.

The KS10 Central Processing Unit

The KS10 central processing unit consists of four extended hex modules. They are:

- The Data Path Executive (DPE) Module that contains data path, registers, cache, PI system, and Dispatch ROM (Read-Only Memory)
- The Data Path Memory (DPM) Module that contains bus interface, processor status flags, paging, cache directory, and shift counter
- The Control RAM Address (CRA) Module that contains next microcode address logic, microcode leading hardware, and $2K \times 36$ bits of microcode
- The Control RAM Memory (CRM) Module that contains $2K \times 60$ bits of microcode.

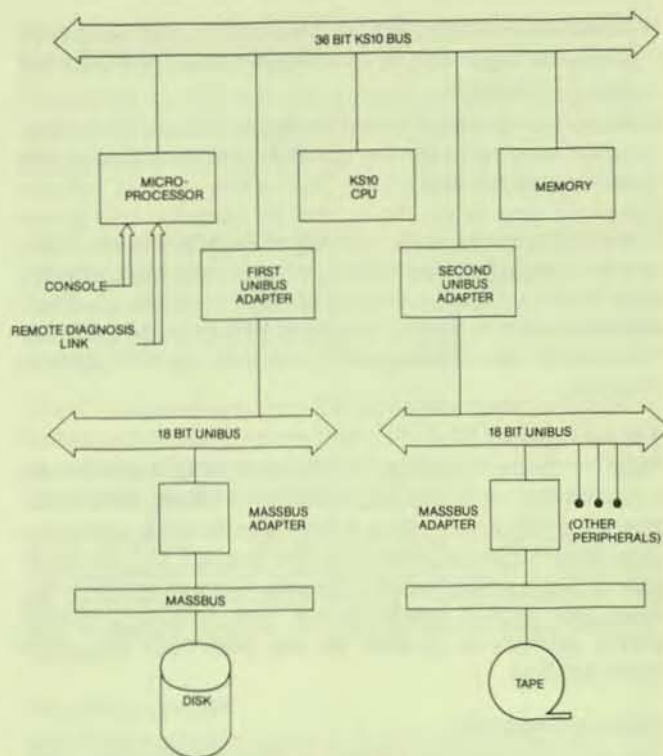


Figure 6-1
System Architecture

The heart of the KS10 is an internal synchronous back-plane bus, called the KS10 bus. A multiplexed, two-cycle bus, it provides a control and data path between the processor, memory, console, and peripheral devices. Command and address information is transmitted from one bus device to another during the first 150-nanosecond bus cycle. Data is then transferred between the devices during the following bus cycle.

The bus operates in a bus request/bus grant mode — before any device can transfer information, it must first request and be granted the bus. There is a bus request line and a corresponding grant line for each device. The bus arbitrator on the console module monitors all requests, resolves priority, and, whenever the bus is free, grants the bus by asserting the grant line for the requesting device with the highest priority. The bus arbitrator can handle up to eight devices.

Figure 6-1 illustrates the connection of the KS10 central processor, memory, UNIBUS Adapters, and console via the KS10 bus.

The bus performs several major functions:

- Transfer data to and from MOS memory through the memory controller under control of the CPU, console, or a UNIBUS Adapter.
- Transfer data to and from I/O device registers under control of the CPU or console. Any device outside the CPU is considered an I/O device.
- Transmit priority interrupt requests generated by the UNIBUS Adapters and, upon CPU request, provide device numbers and interrupt vectors from the interrupting device to the CPU.

- Provide a continuous clock train used by all devices to sequence logic and to synchronize operation with the rest of the system.
- Equip the console to reset the system, to perform diagnostic functions, and to signal ac power failure to the devices on the bus.

The KS10 bus data path is 36 bits wide. Additionally, there are two parity bits associated with the data lines, one for data lines 0 - 17 and a second for data lines 18 - 35. Each device checks for correct parity when it receives information over the bus. If bad parity is detected, the CPU clock is stopped.

Cache Memory

Cache memory decreases instruction execution time by substantially speeding the average memory reference. Main memory access time is 900 nanoseconds; cache access time is 300 nanoseconds. The typical KS10 cache hit rate of 80 percent gives an effective average memory access time of 420 nanoseconds. The KS10's 512-word cache memory is located on the Data Path Executive (DPE) Module.

General Registers

The general registers consist of eight accumulator (AC) blocks of 16 words each, located on the DPE Module. The eight sets of registers can be used as accumulators, index registers, or the first 16 locations in memory. How the registers are used depends upon how they are addressed in the instruction word. Access time to general registers is 300 nanoseconds.

Microstore

The microstore is located on the Control RAM Address and Control RAM Memory Modules. 2,048 96-bit words are provided. The microcode and microstore are considered integral to the hardware and are not available for operating system or user program use.

Instruction Set

The KS10 central processing unit features 396 microprogrammed instructions. Each instruction is a series of microinstructions that performs various logical functions such as processor state control, data path control, and actual implementation of each instruction in the KS10's set. Since the set provides so many instructions, fewer instructions are typically required to perform a given function. The instructions are logically classed and consistent in mode, making their use straightforward.

All instructions are capable of directly addressing 256K words without resorting to base registers, displacement addressing, or indirect addressing. Instructions can use indirect addressing with indexing to any level. Many instruction classes, including floating point, allow immediate mode addressing where the result of the effective address calculation is used directly as an operand to save storage and speed execution.

In all instructions, the nine high-order bits 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 always

supplies information for calculating the effective address. The effective address can be used as an operand (immediate mode) or as the actual (virtual) address to fetch an operand or alter program flow. Bit 13 specifies the type of addressing (direct or indirect); bits 14 - 17 specify an index register used for address modification (zero indicates no indexing). The remaining 18 bits contain a memory address.

Half-Word Data Transmission - Half-word data transmission instructions move half a word and can modify the contents of the other half of the destination location. The 16 sets of instructions differ in the direction they move the chosen half-word and in the way they modify the other half of the destination location.

Full-Word Data Transmission - Full-word data transmission instructions move one or more full words of data. The instructions can perform minor arithmetic operations, for instance, calculating the negative or the magnitude of the word being processed.

Byte Manipulation - Five byte-manipulation instructions pack or unpack bytes of any length anywhere within a word.

Logic Instructions - Logic instructions shift, rotate, and execute the 16 Boolean operations on two variables.

Fixed-Point Arithmetic - The KS10 is a 2's complement machine in which zero is represented by a word containing all zeros. As in comparable implementations, the KS-10 logic does not keep track of a binary point. The programmer must adopt a point convention and shift the magnitude of the result to conform to the convention used. Two common conventions are to regard a number as an integer (binary point to the right) or as a proper fraction (binary point to the left). In these cases, the range of numbers represented by a single KS-10 word is -2^{35} to $2^{35}-1$ (binary point to the right) or -1 to $1-2^{-35}$ (binary point to the left). Since multiplication and division use double-length numbers, there are special instructions with integral operands for these operations.

The format for double-length, fixed-point numbers is an extension of the single-length format. The magnitude (or its 2'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; bit 0 of the low-order word is ignored. The range for double-length integer and proper fractions is thus -2^{70} to $2^{70}-1$ (binary point assumed at the right) or -1 to $1-2^{70}$ (binary point assumed at the left). Zero is represented by a word containing all zeros.

Floating-Point Arithmetic - The KS10-E processor has instructions for scaling, negating, adding, subtracting, multiplying, and dividing in single- and double-precision floating-point format.

In single-precision floating-point format, one bit is used for the sign, eight bits for the exponent, and 27 bits for the fraction. In double-precision floating-point format, one bit is used for the sign, eight bits for the exponent, and 62 bits for the fraction. This results in a precision in the fraction of 1 part in 4.6×10^{18} and an exponent of 2 in the range from -128 to $+127$.



Normalized single-precision floating-point numbers have a fraction that can range in magnitude from $\frac{1}{2}$ to $1 - (2^{-27})$. Increasing the length of a number to two words does not significantly change the range, but it does increase the precision. In any format the magnitude range of the normalized fraction is from $\frac{1}{2}$ to 1 less the value of the least significant bit. In all formats the exponent range is from 2^{-128} to 2^{127} .

Fixed/Floating Conversion — Special KS10 instructions convert fixed-point formats to or from floating-point formats. Two sets of instructions perform this function, one set optimized for FORTRAN and a second set optimized for ALGOL.

Arithmetic Testing — The arithmetic testing instructions can cause a program jump, no jump, skip, or no skip, depending on the result of an arithmetic test, and may first operate arithmetically on the test word.

Logical Testing, Modification, and Skip — A group of instructions modify and/or test, using a mask. They cause a program skip or no skip, depending upon the value of selected bits in an accumulator.

Program Control — Program control instructions include several types of program jump instructions and subroutine control instructions that jump as well as save or restore information on a stack.

The KS10 instruction set includes a group of operation codes that are used as monitor calls. When included in a user program, these instructions trap to the operating system, which examines the instruction and takes the appropriate action. This monitor call implementation is fundamental to TOPS-20 operation because in a multitask environment, the operating system must control certain operations.

Business Instruction Set — There are five classes of instructions in the Business Instruction Set of the KS10 central processor. Four of these are arithmetic instructions to

add, subtract, multiply, and divide, using double-precision, fixed-point operands.

The STRING instruction, the fifth class, performs nine separate string functions. These functions include an edit capability; decimal-to-binary and binary-to-decimal conversion in both offset and translated mode; and move string and compare string in both offset and translated mode. Offset mode is byte modification by addition of the effective address of the string instruction. Besides providing the translation function, these instructions can control AC flags and can detect special characters in the source string.

The Business Instruction Set provides faster processing because there are special instructions for a range of string operations. These instructions can be used on a variety of code types, such as ASCII and EBCDIC.

Trap Handling — The KS10 provides programmed trap instructions to directly handle arithmetic overflows and underflows, pushdown list overflows, and page failures. This trap capability avoids recourse to the program interrupt system.

Processor Modes

Instructions are executed in either Executive Mode or User Mode, depending on the state of the mode bit. In Executive Mode operations, all implemented instructions are legal. The monitor operates in Executive Mode and is able to control all system resources and the state of the processor. In User Mode certain instructions, such as direct I/O, are illegal and cause a trap to the monitor. Users are required to issue monitor calls for such system services.

MEMORY

The KS10 central processor supports a hierarchical memory system consisting of fast register blocks, high-speed cache memory, main (MOS) memory, and mass memory subsystems such as disks. The fast register blocks, cache memory, and main (MOS) memory are directly controlled by the KS10 hardware and firmware. Accessing is microstore controlled. Disk memory access is controlled by the operating system.

Memory Address Mapping

The memory address hardware was developed in conjunction with the software so that memory management is totally transparent to the user. Physical memory is divided into 512-word segments called pages. All addresses, both monitor and user, are translated from the program's virtual address space to the physical memory address space. This facilitates protection of the monitor and efficient use of physical memory. For example, only a portion of the monitor need be permanently resident in memory, resulting in more memory available to the user.

The high-order nine bits of an 18-bit virtual address constitute the virtual page number and are used to index into a hardware page map. If the 13-bit physical page number is found in the hardware page map, then the low-order nine bits of the virtual address are appended to the 13-bit physical page number to form the complete physical memory address. If the entry in the hardware page map

does not exist, the memory processor gets the entry from the individual page map in memory, and updates the hardware page map.

There are separate page maps in memory for the monitor and for each user. These maps contain storage address pointers that identify a page either in memory or on disk. The system interprets three types of address pointers.

The first, called a private pointer, contains a physical storage address. If this is a memory address, the system loads the hardware page table with the information and completes the requested reference. If it is any other address, the system initiates a trap to the monitor for appropriate action.

The second, called a shared pointer, contains an index into a system table at a fixed location. This system table is called the Shared Page Table (SPT) and contains the physical storage address of the shared page. The table, addressed through a hardware register, enables the software memory management routines to maintain just one pointer to a page, no matter how many processes are sharing the page.

The third, called an indirect pointer, contains a page number and an SPT index. The SPT index is used to pick up an address of a page table. To obtain the physical storage address, the page table is indexed by a page number given in the indirect pointer. The pointer allows one page to be exactly equivalent to another page in a separate address space.

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

MOS Memory

The main memory subsystem of the KS10 central processor consists of metal oxide semiconductor (MOS) memory connected to the KS10 bus by a memory controller. The MS10 memory is available in 64K word increments up to a maximum of 512K words. The memory word length is 43 bits: 36 data bits and seven error-detection and correction bits. Memory interleaving is not required.

Semiconductor memory offers the advantage of high speed, especially over core memories. It also offers ease of interfacing to control "outside world" electronics. The processes used to construct the storage elements within the memory array itself can also be used for addressing and decoding electronics and for output buffering — on a single chip. Fewer connections mean higher reliability and lower cost.

In the MOS memory implementation, the storage element takes the form of a MOS "cell" or transistor. In an elementary operation, data bits are stored by charging or not charging the effective capacitance of each storage cell. Reading entails sensing the presence or absence of charge. Only a few hundred electrons of charge differentiate the storage of a 1 and a 0. Because there is capacitive leakage, this charge must be periodically renewed or refreshed. The required frequency of the refresh cycle depends on leakage of the circuit, which increases with tem-

perature. The 128 rows of KS10 MOS memory are automatically refreshed by the memory hardware, one row every 15 microseconds.

All information stored in MOS memory is lost if power is removed from the memory circuitry.

MS10 memory

The MS10 is a single-port memory using MOS random access memories (RAMs). It corrects single-bit errors and detects double-bit errors. If the memory control logic detects a correctable error on a memory read cycle, the following occurs:

- If the error is in one of the 36 data bits, the error bit and the parity bit are automatically corrected
- If the error is in one of the seven check bits, no correction is required, an error flag is raised, and the address of the failing memory is held in a register
- A flag is raised indicating that the logic has corrected an error
- The memory control logic saves a 20-bit address (for a read or read-pause-write request) and the ECC code for the first error detected to allow software monitoring of memory condition

The MS10 memory is highly reliable because:

- Single-bit error detection and correction and double-bit error detection are implemented in the memory control logic
- Power supply is a high-efficiency switching regulator for low internal power dissipation
- Storage array component layout facilitates airflow over module
- Extensive airflow and temperature profile measurements have been taken
- Modules are "burned in" prior to installation
- Faults are isolated to board level
- Correctable read errors are automatically handled by the hardware, so correct data is always sent to the CPU and notification of the corrected error is provided

CONSOLE SUBSYSTEM

The console is an extremely important subsystem in the KS10 central processor because it controls all console and diagnostic functions. The console is housed on a single, extended hex module and uses an eight-bit microprocessor. To program the microprocessor, an 8 KB programmable read-only memory (PROM) and 1KB random access memory are provided. It also has two universal synchronous/asynchronous receiver/transmitter (USART) interfaces, one for console operation and one for remote diagnosis link operation.

The console is interfaced to the KS10 bus for data and control functions. In addition, direct connections to other KS10 CPU modules allow the console to perform test and housekeeping functions.

The local operator controls the KS10 with a set of commands typed at the console terminal (CTY). The CTY connects directly to the microprocessor-based console hardware over a serial line interface (USART). A second serial

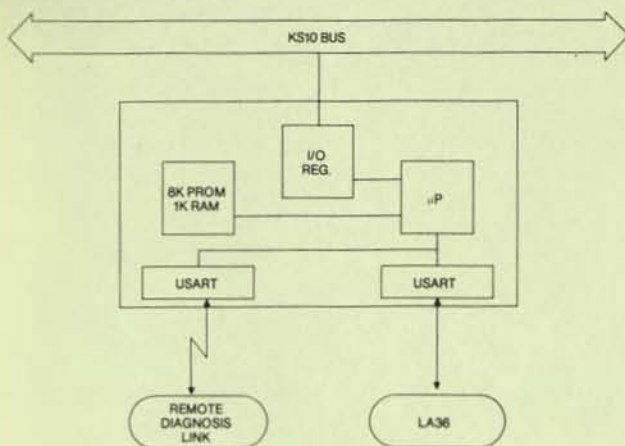


Figure 6-2
KS10 Console Subsystem

line, operating in parallel with the first, can also be connected to the console hardware to control the KS10 by a remote diagnosis link.

Commands typed at the CTY or entered from the remote diagnosis link are implemented by a program running in the console's microprocessor. This program, resident in PROM, automatically runs when power is turned on, and is not destroyed when the system is turned off.

The CTY and remote diagnosis link can be operated in either user mode or console mode. Commands are the

same in either local or remote CTY cases, except that the remote link repertoire is restricted. In user mode, the CTY is a user terminal, and commands are passed to and from the KS10 CPU under control of the console program. An exception is typing a "control \ " which causes the console program to switch the CTY from user mode to console mode. All other commands performed in user mode are a function of the operating system.

In console mode, commands are directed to an executive by the microprocessor console hardware. An operator can:

- Reset and bootstrap the system
- Load and check the KS10 microcode
- Deposit and examine memory
- Read and write I/O device registers
- Transmit to and receive from the KS10 backplane bus
- Start and stop the CPU clock
- Single-step the CPU clock
- Execute a given instruction
- Halt the machine
- Start the machine at a given location

The console program initializes the CTY to console mode at powerup. When in console mode, either starting execution, continuing execution, or typing a "control z" will cause the CTY to switch to user mode. Also, an error that lights the fault indicator will cause a return to console mode, as will any KS10 processor halt instruction.

7 The Peripherals



7
The
Peripherals

DECSYSTEM-20s support a family of peripheral devices that includes disks, magnetic tapes, terminals and terminal interfaces, lineprinters, cardreaders, and a papertape reader/punch.

The wide selection of peripheral devices reduces the storage burden on disks and provides convenient media for file archives. Infrequently used programs or data files can be stored on magnetic tape, which is easily transported and accessed. A cardreader and a papertape reader/punch are also available for this type of I/O. Using multiple lineprinters with different characteristics can improve hardcopy output.

The terminals supported by the DECSYSTEM-20 are the hardcopy LA36, LA37, LA38, LA120, and the VT100 video terminals.

A front-end processor that supports synchronous communications is available for DECSYSTEM-20s.

The DN200 remote station is discussed under the heading *Remote Job Entry Stations* in Section 10, *Communications*.

DECSYSTEM-20s support four types of peripheral systems:

- Mass storage peripherals (disks and magnetic tapes)
- Unit record peripherals (lineprinters, cardreaders and papertape devices)
- Terminals and terminal line interfaces
- Communication front-end processors

All byte capacities in this section are calculated in 8-bit bytes.

Peripheral devices for the KS10 (DECSYSTEM-2020) processor are UNIBUS and MASSBUS devices that interface to the system through UNIBUS Adapters (UBAs). The UBA is a single extended hex module connecting to the KS10 backplane bus and a UNIBUS. Two UBAs are standard in the DECSYSTEM-2020. One UBA and one UNIBUS are reserved for disks. The second UBA and UNIBUS are used for all other devices: tape, lineprinter, synchronous communications, and asynchronous communications lines.

Details of the KL10-E (DECSYSTEM-2040, 2060) I/O subsystem can be found in Section 5 under *Input/Output Subsystem*.

MASS STORAGE PERIPHERALS

The DECSYSTEM-20 mass storage peripherals are moving-head disk drives and magnetic tape transports.

Disks

Disk features include accurate servopositioning, error correction, and offset positioning recovery. Table 7-1 summarizes the capacities and speeds of the disk devices.

To support the performance and reliability features of the disk devices, the operating system's disk device drivers provide comprehensive error-recovery algorithms (e.g., ECC and offset recovery for disk) and log all device errors.

Disk controllers allow several drives to perform simultaneous seek operations. Since the controller is not busy during seek operations, data transfers on one drive can overlap seeks in progress on other drives. If more than one drive is on the same controller, overlapped seeks will accelerate processing for disks that are on that controller. Overlapped seeks increase throughput by decreasing the effective access time.

RM03 Disk-Pack Subsystem — The RM03 is a top-loading, free-standing disk drive housed in a dedicated cabinet. Subsystems are expandable to eight disk drives or 536 MB. The RM03 has a formatted capacity of 67 MB and an average access time of 38 milliseconds. Its maximum data transfer rate is 250 thousand 36-bit words per second. The disk does 18-bit NPR data transfers over a UNIBUS and 36-bit NPR data transfers over the KS10 backplane bus (see Section 6). The RM03 is available only with KS systems.

RP06 Disk-Pack Subsystem — The RP06 is a large-capacity disk drive offering both high performance and a low price per megabyte of storage. It uses a removable disk pack and has a variety of large-disk features.

RM03 and RP06 disk drives can be mixed on the same RH11 controller on a KS10; however, they cannot be mixed within the same logical disk structure.

RTP20 Disk Subsystem — The RTP20 has the largest capacity of any disk offered by DIGITAL — 929 MB per RP20 disk unit. It consists of a controller and an RP20 disk drive, and can support two additional RP20s.

Each RP20 data module has fifteen recording surfaces with two read/write heads per surface and a transfer rate of 1.2 MB per second. With the optional dual-port feature, each data module can independently transfer data. The RP20 is available only with KL systems.

Tape Devices

DECSYSTEM-2020s are available with TU77 magnetic tape drives. Other DECSYSTEM-20s are available with this drive plus the TU70 series tape subsystems. All provide capacities and speeds needed for a wide range of DECSYSTEM-20 applications.

All tape devices use a mylar-based, oxide-coated magnetic tape with reflecting marker strips to indicate Beginning of Tape and End of Tape. Adjacent files are separated by formatted interrecord gaps. In addition, industry-compatible formats facilitate data transfer among computers and can reduce hardware costs.

All tape devices provide write-protection for data that is read and written onto the tape. An industry-standard write-protect ring is located on the tape reel. The tape drive can sense write-protection and prohibit data writing.

Accurate data recording and retrieval is ensured on all tape systems with read-after-write checks. This check verifies that proper data is written on the tape, eliminating the chance of data being written on worn or damaged sections of tape. If an error is detected in the read-after-write check, a message is sent to the processor.

Parity checks and longitudinal (LRC) and cyclic redundancy checking (CRC) further ensure the accurate transfer of data in all magnetic tape systems. Parity is checked character by character when reading and writing on tape at 6250 (GCR), and 1600 (PE) bits per inch. 800 (NRZI) bits-per-inch operation also includes a CRC character and an LRC character. CRC and LRC characters are calculated when a block is written and checked when the block is read. If an error is detected, an indication is made to the host computer.

All magnetic tape systems minimize bad-tape-error problems through a runaway timer that allows the system to recover from bad tape sections on the reel.

Magnetic tape controllers/formatters perform similar operations. These include:

- Moving the tape forward or backward to a new position
- Monitoring tape operation
- Fetching, formatting, and sending data
- Handling error conditions and drive-servicing requirements

TU70 Series Magnetic Tape System — The TU70 series is well suited for high-speed recording applications such

as journaling of transaction data and backup of large disk volumes. It is a fully integrated, high-performance system specifically designed to operate with the DECSYSTEM-20.

A system consists of one or more TU72-E 9-track drives, plus a maximum of eight drives, a TX02 tape control unit, and a DX20 data channel. The tape control unit connects to each drive with radial cabling, and the DX20 data channel connects directly to the internal data channel of the DECSYSTEM-20.

When reading and writing in PE mode, the accuracy of data transfer is confirmed with character-by-character parity checking. Error correction for single-track dropout is dynamic. The GCR mode uses a polynomial error-detection scheme on data blocks within each record. Data reliability is increased with a self-clocking feature that is independent of tape skew.

TU77 Magnetic Tape System — The TU77 is a high-performance tape storage system that is also suitable for many high-duty-cycle applications such as backup from disk to tape and transaction processing. Design considerations, such as eliminating mechanical relays and incandescent lamps on the drive and using air bearings and ceramic guides for reduced media wear, help ensure its reliability. A tape interlock disables tape motion if there is a pressure loss in the vacuum system. This feature reduces the chance of accidental tape damage.

The TU77 is a fully integrated system that is packaged in a dedicated cabinet with its associated interface and power supply. The TU77 subsystem consists of a TU77 tape drive, a dual-density TM03 tape formatter, and a MASSBUS controller that mounts in the processor chassis. Each subsystem can include up to four tape drives with a total maximum of 16 drives per DECSYSTEM-20.

Automatic tape threading maximizes operator convenience and minimizes tape handling. Smaller reels of tape can be threaded automatically when tape is placed manually in the loading slot. If a 10.5-inch reel of tape fails to load on the first attempt, the TU77 will detect the misfeed and reload without operator intervention.

The TU77 subsystem also promotes data integrity through automatic correction of single-track errors. In PE mode, this error correction is done automatically by the hardware. In NRZI mode, error correction is performed under software control.

The TU77 reads in forward and reverse and uses the industry-standard data format.

UNIT RECORD PERIPHERALS

The DECSYSTEM-20 supports a full line of unit record peripherals including lineprinters and card and papertape devices.

LP20-A and -B Lineprinters

The LP20-A and -B lineprinter systems have two hardware components: an LP05 lineprinter and an LP20 lineprinter controller and data source interface.

The LP05 lineprinter is a medium duty-cycle drum lineprinter. It is designed for data processing environments that require good print quality and moderate print volumes. The LP05 is an impact-shaped (whole) character, 132-column lineprinter. It prints at 300 lines per minute using a 64-character set or at 230 lines per minute using a 96-character set. It performs a single forms (paper) step in 45 milliseconds (maximum) and, when formatting vertically, slews forms at up to 67.8 centimeters per second.

Up to 132 characters can be stored in a print line buffer. Upon command from the data source, the LP05 prints the contents of the buffer and advances the forms as specified. It signals the data source when it is ready for the next line of print data or forms motion command.

The LP05 uses a rotating drum containing all the characters in front of the forms and ribbons. Fifty-eight print hammers behind the forms are timeshared between 132 data columns to produce inked characters and carbon transfer characters on multicopy forms.

The LP05 contains a 12-channel programmable vertical format unit (PVFU). It uses a format memory that is loaded from the data source, so no format tape is installed by the operator and there is no risk of running a job with the wrong format tape. The PVFU can be loaded each time data is requested by the printer. Format memory data and control codes are transmitted to the printer via the normal data lines using the standard demand/strobe communications. The PVFU can control the vertical movement of forms having eight to 143 lines.

The LP05 lineprinter has switches and indicators for operation and operator-level maintenance. When the print TEST/ONLINE switch is set to TEST, the self-test module operates as a built-in data source. The printer communicates with the self-test module on a demand/strobe basis, stores the data received in the line buffers and paper motion registers, and processes the data. After the line of data has been printed and the paper moved, the printer resumes the data exchange communication with the self-test module in a continuous cycle.

Table 7-1 Disk Devices

Disk	Type	Capacity (Mbyte)*	Peak Transfer Rate (Kbyte/s)*	Average Seek Time (ms)	Average Rotational Latency (ms)	Interface	Maximum Drives per Controller
RM03	Removable	67	1,200	30	8.30	MASSBUS	8
RP06	Removable	176	806	30	8.3	MASSBUS	8
RP20	Nonremovable	929	1,200	25	8.3	MASSBUS	4†

* K = 1,024; M = 1,024²

† 2 spindles per drive

A long-line interface provides the LP05 with differential receivers and drivers so that it can be located up to 30.5 meters (100 feet) from the data source. A standard 7.6-meter (25-foot) device cable is normally supplied.

LP20-C and -D Lineprinters

The LP20-C and -D lineprinter systems are composed of an LP14 lineprinter and LP20 lineprinter controller and data source interface.

The LP14 lineprinter is a medium duty cycle drum lineprinter. It's designed for use in data processing environments that require good print quality and medium print volumes. The LP14 is an impact type, shaped (whole) character, 132-column lineprinter capable of printing six or eight lines per inch. It will produce printed output at 890 lines per minute using a 64-character set, or 650 lines per minute using a 96-character set. It performs single forms (paper) step in 20 ms (max) and slews forms at up to 76.2 cm (30 inches) per second when formatting vertically.

The LP14 lineprinter stores streams of up to 132 characters in a print-line buffer, and, upon command from the data source, prints the contents of the buffer and advances the forms as specified by the command. It signals the data source when it's ready for the next line of print data or forms-motion command.

The LP14 contains a 12-channel programmable vertical format unit (PVFU). The PVFU uses a format memory that is loaded from the user system. This relieves the operator from having to install a format tape and eliminates the risk of running a print job with the incorrect format. The PVFU may be loaded at any time data is requested by the printer. Format memory data and control codes are transmitted to the printer via the normal data lines using standard demand/strobe communications. The PVFU can control the vertical movement of forms having from eight to 143 lines.

The LP14 lineprinter includes an operation/maintenance panel that mounts switches and indicators to operate the lineprinter and perform operator-level maintenance.

A self-test module allows the printer to be tested under dynamic conditions. When the printer TEST/ON-LINE switch is set to TEST, the self-test module acts as a built-in data source; i.e., the printer communicates with the self-test module on a demand/strobe basis, stores the data received in the line buffers and paper motion registers, as

appropriate, and processes the data. After the line of data has been printed and the paper moved, the printer resumes continuous cycle data exchange communication with the self-test module.

LP200 Lineprinters

The LP200 lineprinter system has two hardware components: an LP07 lineprinter and an LP20 lineprinter controller and data source interface.

The LP07 is a high-performance, horizontal font motion lineprinter designed for data processing environments requiring high-grade print quality, heavy print volumes, and high reliability.

The LP07 is an impact-shaped (whole) character, 132-column lineprinter. It prints 990 or 1,220 lines per minute using a 96-character set and 715 or 905 lines per minute with a 64-character set. Print speed is operator-selectable. The LP07 does a single forms paper step in 12.5 milliseconds and slews forms vertically at up to 152.4 cm (60 in) per second.

Up to 132 characters can be stored in a print line buffer. Upon command from the data source, the LP07 prints the contents of the buffer and advances the forms as specified. It signals the data source when it is ready for the next line of print data or forms motion command.

The LP07 uses a Charaband* as the horizontal font carrier in front of the forms. One hundred thirty two print hammers, behind the forms and the ribbon, produce the inked characters and carbon transfer characters on multicopy forms. The Charaband has the advantages of train printers, but minimizes the problems of character set rigidity, friction, and wear associated with other horizontal font techniques.

The Charaband has two complete character sets, one on each side. A one-minute manual operation switches the sets.

The LP07 lineprinter contains a direct-access vertical form unit (DAVFU) that provides the same benefits as the PVFU discussed in the LP20 section above. The DAVFU also permits print density of six or eight lines per inch under program control.

* Charaband is a trademark of Dataproducts Corporation

Table 7-2 Tape Devices

Tape	Type	Reel Size (in)	Capacity (Mbyte)*	Column Buffering	Recording Density (bits/in)	Read/Write Speed (in/s)	Maximum Data Transfer Rate (10 ³ char/s)	Rewind Speed (in/s)	Interface	Maximum Drives per Controller
TU72	9-track	10.5	40	Vacuum	1,600/6,250	125	750	500	MASSBUS	8
TU77	9-track	10.5	40	Vacuum	800/1,600 [†]	125	200	440	MASSBUS	4

* M = 1,024²

[†] Program-selectable

The LP07 contains a self-test unit similar in function to the unit discussed in the LP20 section above.

A long-line interface provides the LP07 with differential receivers and drivers so that it can be located up to 152 meters (494 feet) from the data source. A standard 7.6-meter (25-foot) device cable is normally supplied.

Cardreaders

The CD20 cardreaders read encoded, punched information using the American National Standard 8-bit card code and interprets a special punch outside the data representation to indicate end-of-file. The cardreaders use the industry-standard EIA card that has 80 columns and 12 zones, or rows.

The cardreaders are designed to meet different throughput requirements. CD20 cardreaders are available in both table (CD20-A) and console (CD20-C) models. CD20-A can process 285 punched cards per minute, and the CD20-C, 1,200 cards per minute.

The CD20-A tabletop unit has an input hopper capacity of 1,000 cards and the CD20-C, 2,250 cards. They are designed to prevent jams and keep card wear to a minimum. These readers also have a high tolerance for cards that have been nicked, warped, bent, or subjected to high humidity. Readers use a short card path, with only one card in the track at a time. They all use a vacuum-pick mechanism that keeps cards from sticking together by blowing a stream of air through the bottom half-inch of cards in the input hopper.

The CD20-C console cardreader has a single-piece read station with infrared light-emitting diodes, emitters, and phototransistor detectors. No adjustments are required in the ten-year life expectancy of the diodes.

The console model also provides high data integrity with a "resync/error detection" feature. The data strobe can be resynchronized for each data column. The reader will either correctly read a misregistered card or reject the card by halting with a "read check" indication.

Cards can be loaded or unloaded while the console model is operating. A switch can be set to provide either blower shutdown or continued running after the last card has been read. Automatic shutdown reduces computer room noise levels and can also signal the operator that the card hopper is empty.

PC20 Paper Tape Reader/Punch

The PC20 reads eight-channel papertape at 300 lines per second and punches at 50 lines per second in either alphanumeric or binary. It automatically fan-folds the paper tape.

The PC20 contains a photoelectric papertape reader and an electromechanical punch. A set of photodiodes translates the presence or absence of holes in the tape to 1s and 0s.

In alphanumeric mode, a single tape-moving command reads all eight channels from the first line encountered. In binary mode, the device reads six channels from the first six lines in which hole 8 is punched and then assembles the information into a 36-bit word. The PC20 interface con-

tains a 36-word buffer from which all data is retrieved by the processor.

At 300 lines per second, the reader takes 3.33 milliseconds per alphanumeric character and 20 milliseconds per contiguous word.

TERMINALS AND INTERFACES

A KS system supports a maximum of 32 local and remote line interfaces. A KL system supports up to 128 local and remote lines. Programs can control terminal operations through the terminal driver. The terminal driver receives and services interrupts, initiates I/O operations, cancels in-progress I/O operations, and performs other device-specific functions. A program can:

- Get data from an open terminal without stalling program execution. The program doesn't have to wait for incoming data to be available in the terminal input buffer.
- Through echo control mode, using a full-duplex terminal, simulate the operations of a blockmode terminal. Input data from the terminal is processed a field at a time, without affecting the displayed output in other fields on the screen.
- Translate, interpret, and transmit ESCAPE sequences.

In addition, pseudoterminals can be used for jobs, such as batch, that do not require operator intervention. A pseudoterminal has the characteristics of a terminal but has no physical device attached to it. Like a terminal, a pseudoterminal has both input and output buffers. By using pseudoterminals, one job can control other jobs on the system.

Terminal characteristics are initially established in a command file during software installation. However, system users also can modify the characteristics of their particular terminal. For example, users can:

- Set the width of the print line between 1 and 255 columns. The system automatically generates a carriage return and linefeed sequence after the specified number of characters has been typed or printed.
- Control transmission of upper- and lowercase characters.
- Allow the computer to interrupt transmission of characters from the terminal (XOFF) or instruct the terminal to resume transmission of characters (XON). The terminal hardware must be present to respond to XOFF, a XON characters.
- Set the rate at which the terminal's interface can accept or pass characters.
- Set even, odd, or no parity checking.

A line entered on a command terminal is terminated by any of several special characters. The RETURN key, for example, is one of the most common means of transmitting a command to the host. A program reading from a terminal can specify a particular line terminator for read requests.

LA120 Hardcopy Terminal

The LA120 is a hardcopy terminal with high throughput and a number of advanced keyboard-selectable formatting and communication features. It uses a contoured,



typewriter-style keyboard, an additional numeric keypad, and an LED display for terminal characteristics.

Several features give the LA120 its high throughput:

- 180-character-per-second print speed
- 14 data transmission speeds ranging up to 9,600 baud
- 1,024-character buffer to equalize differences between transmission speeds and print speeds
- Smart and bidirectional printing so that the printhead always takes the shortest path to next print position
- High-speed horizontal and vertical skipping over white space

In addition to its throughput, the LA120 is distinguished by its printing features. The terminal offers eight font sizes ranging from expanded (five characters per inch) to compressed (16.5 characters per inch). Hence, a user could select a font size of 16.5 characters per inch and print 132 columns on an 8.25-inch-wide sheet. Other print features include six line spacings ranging from two to 12 lines per inch, user-selectable form lengths up to 14 inches, left/right and top/bottom margins, and horizontal and vertical tabs.

The LA120 is designed for easy use. Terminal characteristics are selected via clearly labelled keys and simple mnemonic commands. Once the selections have been made, the operator can check the settings by depressing the STATUS key. The terminal will then print a listing of the selected settings.

LA38 Hardcopy Terminal

The LA38 DECwriter IV is a low-cost, desktop microprocessor-driven terminal capable of processing data at up to 30 characters per second. The LA38 includes a universal power supply, a standard EIA interface and EIA null modem cable, an 18-key numeric keypad, paper-out switch, paperfeed tractor, and user-assistance documentation package.

Features such as horizontal tabs and margins, and a choice of four character sizes and six line spacings can be

set and changed by the host computer or by the user. The LA38 has a permanently stored format for a computer printout. When the terminal is powered up, it automatically is set to print ten characters per inch and space six vertical lines per inch, set tab stops every eight spaces, set the left margin at column 1 and the right margin at column 132.

The LA38 operates at 300 baud and can print at burst speeds up to 45 characters per second. An alternate speed of 110 baud and 10 characters per second can be selected from the keyboard. The desktop configuration and sculptured typewriter-like keyboard are so similar to standard typewriters that the transition from typewriter to terminal is natural.

The terminal's basic design contributes to its reliability and maintainability. Logic and power amplifier on a single board with custom LSI reduces the component count and increases circuit reliability.

The LA38 senses printhead jams instantly and shuts down power to the printhead drive until the jam is corrected and the terminal restarted. This prevents motor overloads and blown fuses. The highly reliable printhead has been designed and tested to print more than 100 million characters. The printhead can be adjusted to adapt to various form thicknesses. The LA38 is equipped with a paper-out sensor that, when enabled, implements one of four selectable actions when there is a paper fault.

Although the LA38 has no scheduled preventive maintenance, should a problem occur, the unit disassembles simply and quickly for easy access to all components. Printing self-test diagnostics allows quick and accurate identification of any faulty components. With the new snap-in ribbon cartridge, users can change ribbons quickly and easily.

Wherever possible, ANSI standard escape sequences are used. These same escape sequences are also implemented on DIGITAL's LA120 and VT100, ensuring compatibility among DIGITAL's terminal products.

VT100 Video Terminal

The VT100 video terminal is an uppercase/lowercase ASCII terminal that offers a variety of user-controllable character and screen attributes. The VT100 features a typewriter-like detachable keyboard that includes a standard numeric/function keypad for data entry applications. Also featured are seven LEDs, four of which are program-controlled, that can be used as operator information and diagnostic aids.

The VT100 offers a number of advanced features. The most important of these are:

- Two screen sizes — 24 lines by 80 columns or 14 lines by 132 columns
- Line-by-line selectable characters of regular size, double width and single height, double height and single width, or double width and double height
- Smooth scrolling and split-screen capability
- Keyboard selectable and storable baud rates, tabs, and Answer Back messages
- Special line-drawing graphic characters for business or laboratory applications

- Selectable black-on-white or white-on-black characters on a full-screen basis

Several options further extend the capabilities of the VT100. These include the advanced video option that adds selectable blinking, underline, and dual-intensity characters to the existing reverse video attribute, and an additional RAM module that enables 24 lines of 132 characters.

DZ11 Terminal Line Interface (KS systems)

The DZ11 is a serial line multiplexer whose character formats and operating speeds are programmable on a per-line basis. A DZ11 connects the UNIBUS with up to eight asynchronous serial lines. Each line can run at any of 15 speeds.

Local operation with EIA terminals is possible at speeds up to 9,600 baud. The DZ11 can be used with dialup full-duplex terminals that operate through most industry-standard modems running at 300 or 1,200 bits per second. The DZ11 optionally generates parity on output and checks parity on input. Incoming characters are buffered using a 64-character silo buffer; outgoing characters are processed on a programmed interrupt request basis.

DH11 Programmable Asynchronous Serial Line Multiplexer (KL systems)

The DH11 multiplexer connects the UNIBUS with 16 asynchronous serial communications lines. The DH11 is a high-performance unit with programmable character formats and operating speeds. Data is buffered on input and is transferred directly from memory.

Program-selectable parameters include transmission speeds up to 9,600 baud, character size, stop-code length, transmission mode, and parity generation and checking. An optional modem control multiplexer is available for connecting the DH11 to an autoanswer data set.

As many as eight DH11s can be placed on a single processor for a total capacity of 128 lines.

DL11 Serial Line Asynchronous Interfaces

The DL11 is an interface between a single asynchronous serial communication channel and the processor. It performs serial-to-parallel and parallel-to-serial conversion of serial start/stop data with a double-character-buffered MOS/LSI circuit called a UART (universal asynchronous receiver/transmitter). This 40-pin, dual inline package has all the circuitry necessary to double-buffer characters in and out, serialize/deserialize data, select character length and stop code configuration, and present status information about the unit and each character.

With a DL11 interface, a DECSYSTEM-20 can communicate with a local terminal, such as a console teleprinter, and with a remote terminal via data sets and private line or public switched telephone facilities.

Users can select from 13 standard data rates up to 9,600 baud or can select a nonstandard rate. With most of the standard rates, the interface can offer split-speed operation for faster, more efficient handling of computer output. In addition, character size is strap- or switch-selectable, and parity checking and stop code length are selectable.

DUP11 Single Synchronous Line Interface (KS systems)

The DUP11 is a character-buffered, synchronous, serial line interface capable of two-way simultaneous communication. The DUP11 translates between serial data and parallel data. Output characters are transferred in parallel from the PDP-11 UNIBUS into the DUP11 where they are serially shifted to the communication line. Input characters from the resident modem are shifted into the DUP11 and made available to the processor on an interrupt basis.

The self-contained unit is capable of handling a variety of protocols. These include byte-oriented protocols, such as DDCMP and BSC, and bit-oriented protocols, such as SDLC, HDLC, and ADCCP. Signals needed to establish communications with the Bell Series 200 synchronous modems are resident in the DUP11's Receive Status Register. The DUP11 can transmit data at up to 9,600 bits per second (limited by modem and data set interface level converters).

Its capabilities make the DUP11 well suited for remote batch, remote data collection, remote concentration, and network applications. In addition, multiple DUP11s can be used in applications requiring several synchronous lines.

KMC11-A Auxiliary Processor (KS system)

The KMC11-A is an auxiliary processor, complete with memory, that interfaces to the UNIBUS. The KMC11 improves the performance of the DECSYSTEM-20 computer systems by performing time-consuming system functions in parallel with the host CPU, thereby offloading it. It is especially suited to controlling I/O operations, such as data communications and analog I/O, that require extensive intelligence.

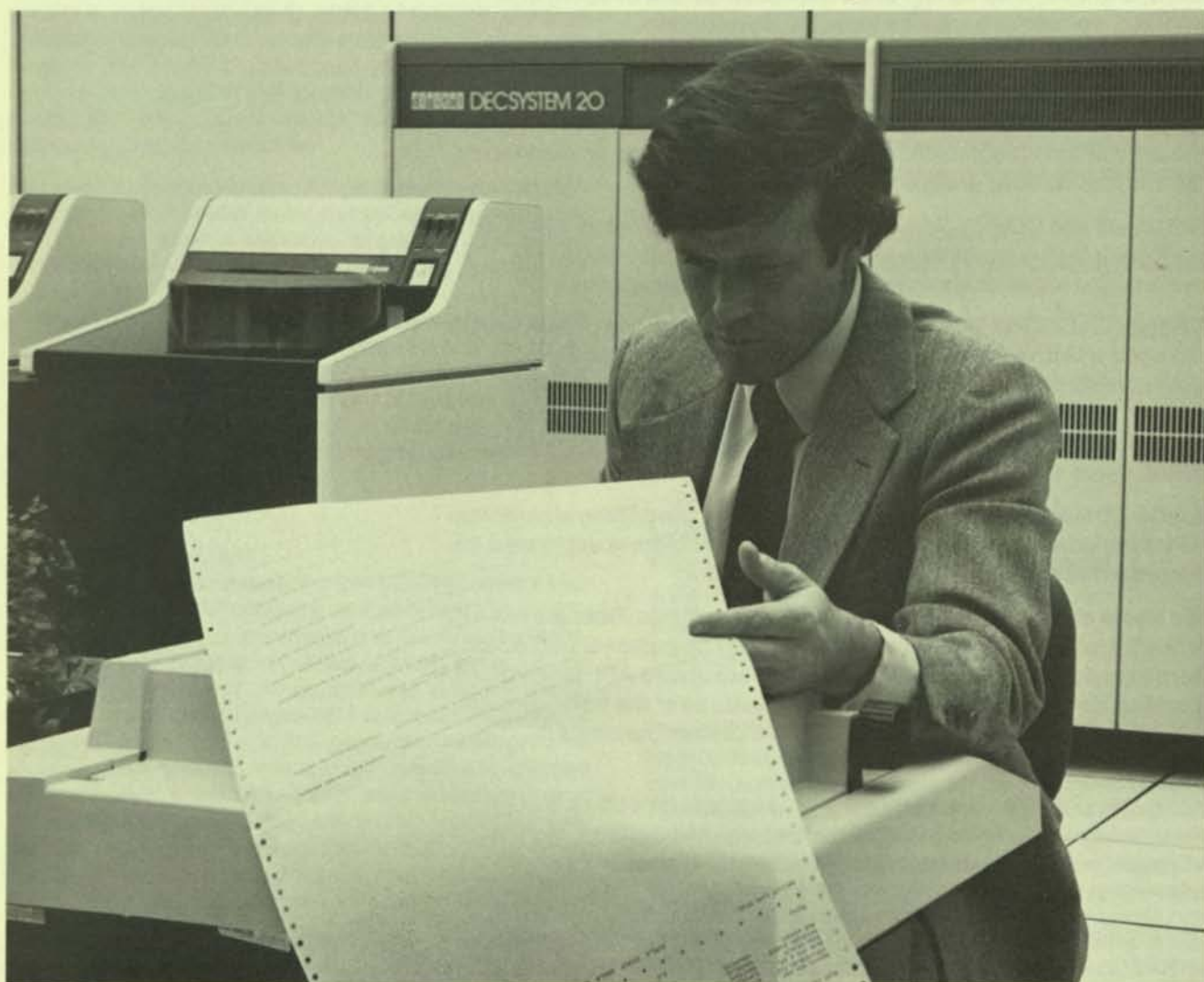
The KMC11 is a high-speed MSI microprocessor (300 nsec instruction time) that uses 16-bit microinstructions and operates on 8-bit data paths. Its 1,024 16-bit word writable control memory that contains the microprogram is loaded by the host processor. A 1,024 8-bit byte data memory stores frequently used information for high-speed access by the microprogram. An NPR UNIBUS interface provides access to control, status, and data registers of one or more peripherals on the UNIBUS. This enables low-cost programmed I/O devices to operate as if they had an intelligent DMA capability.

The KMC11 can be directly connected to a high-speed peripheral such as a DMC11 synchronous line unit. The full-duplex 8-bit parallel interface is well suited to custom designed interfacing.

DN20 COMMUNICATIONS FRONT-END

The DN20 is designed to handle all network communication functions. The DN20 is referred to as a communication front-end to distinguish it from the console front-end that controls the local command terminals and unit record peripherals. The DN20 and console front-end processor communicate with the KL processor through the DTE hardware interface. The DTE resolves the differences between the 36-bit KL word and the 16-bit DN20 word. In systems with the KS10 processor, the communication software resides in the processor.

8 The Languages



In addition to the MACRO assembly language, the TOPS-20 system software supports the optional high-level programming languages FORTRAN, COBOL-68, COBOL-74, ALGOL, BASIC-PLUS-2, APL, BLISS, and CPL. This variety gives the user the most effective or familiar programming solution to match the problem.

FORTTRAN-20 is both a globally optimizing compiler and a runtime system with an interactive debugger and subroutine library. It has been designed to simplify the programmer's job, provide superior compile-time and runtime diagnostics, facilitate debugging, and produce error-free and fast-running programs.

COBOL-68 and COBOL-74 are high-level implementations designed specifically for business data processing. COBOL can be used to create online terminal applications or to write batch applications.

Although BASIC-PLUS-2 is an ideal language for novice programmers who need a fast, easy way to solve problems, it is also a powerful and efficient language suitable for sophisticated applications. BASIC-PLUS-2 is a compiler that produces fast-running programs, yet it retains the highly interactive immediate mode feature of primitive BASIC.

ALGOL-20 is a scientific language designed for describing algorithms. It is a problem-solving language in which the problem is expressed as complete and precise statements of a procedure.

Two levels of the popular APL language are available. Both are full APL implementations with significant extensions. The basic version suits users who do not require the file I/O or the advanced APL function. The extended version, APL-SF, has all of the features of the basic version plus advanced features that substantially increase the range of applications in which it can be used.

BLISS-36 is DIGITAL's implementation language for software development. It contains many of the features of a modern, high-level language, yet it also provides the flexibility and access to hardware of assembly language.

CPL (Conversational Programming Language) is an interpreter similar to PL/I. It is a well documented and easy-to-learn subset of ANSI-1976 PL/I.

TOPS-20 ASSEMBLER

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

- Translating symbolic operation codes 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 MACRO expansion capability to facilitate storage, definition, or user-defined instruction
- Providing a sequential numbered listing with symbols cross-referenced to show where they are defined and where they are used

MACRO programs consist of a series of free-format statements that can be prepared on the user's terminal with one of the system's editing programs. The assembler interprets and processes these statements, generates binary instructions or data words, and produces a listing that can contain cross-reference symbols for ease in debugging the program. MACRO produces device-independent programs. It enables the user to select, at runtime, 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 program listing can go to the lineprinter. More commonly, the source program input and binary output are disk files.

The MACRO assembler contains powerful macro capabilities that enable the user to create new language elements. This 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 permits the expansion and adaptation of the assembler to perform specialized functions for each programming job. In addition, changing the macro definition changes it for every call.

DEBUGGING TOOLS

The online symbolic debugging tool (DDT) enables a user to rapidly check and correct a new program. To do this, the source program is assembled and the resulting binary object file with its table of defined symbols is loaded with DDT. Using command strings, the user specifies breakpoints in the program where DDT suspends execution to accept further commands. This segments the program into executable sections that can be individually checked and corrected. Either before DDT begins executing or at breakpoints, the user can examine and modify the contents of any location in the program. DDT also performs searches, gives conditional dumps, and calls user-coded debugging subroutines at breakpoint locations.

The important feature of DDT is that users communicate with it in terms of the original symbolic location name, rather than the machine-assigned locations.

FORTRAN

FORTRAN-20 is a globally optimizing compiler and a runtime system with an interactive debugger. Both have been designed to simplify the programmer's job, provide superior compile-time and runtime diagnostics, facilitate debugging, and produce error-free, fast-running programs.

FORTRAN-20 is based on the 1966 American National Standard (ANSI) FORTRAN, but it has many extensions to that standard and includes many features from the ANSI 1977 FORTRAN. Both the compiler and object-time system are reentrant (sharable). FORTRAN features include:

- Accepted and powerful features useful to a wide range of technically oriented users
- Computational features matched by full-scale data handling and data management facilities
- Global optimization
- PARAMETER statements that provide symbolic specification of compile-time constraints
- INCLUDE statements to compile source code from more than one file
- N-dimensional arrays
- Optionally generated array bounds checking
- Direct-access I/O capabilities
- Debugger that permits examining and modifying program data, statement-by-statement program tracing, and setting pauses on any statement or routine
- OPEN and CLOSE statements for file specification and control
- ENCODE/DECODE statements
- Boolean operators, including equivalence and exclusive NOR, OR, AND, and NOT
- NAMELIST statement and list-directed I/O that provide format-free input and output operations
- Implied DO loops in I/O and data statements
- Full, mixed-mode arithmetic in expressions
- Octal constants
- Full-word masking operations for all logic functions (rather than a result of just true or false)
- Relational operators
- Error-handling capabilities in I/O statements
- Device independence
- Multistatement lines
- Remarks in statement fields

The FORTRAN object-time system, FOROTS, controls the input/output, format interpretation, and numerical conversion for programs compiled by the FORTRAN compiler. The FORTRAN user can reference any mass storage, unit record, or terminal 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 runtime. The devices corresponding to the specific I/O statements — READ, PRINT, PUNCH, ACCEPT, and TYPE — are also assignable at runtime.

Language Extensions

Powerful FORTRAN IV extensions simplify program coding. Some of the enhancements are:

- Array subscripts — any arithmetic expression can be used. If the value of the expression is not an integer, it is converted to integer type.
- Alphanumeric literals — strings of characters bounded by apostrophes that can be used in place of Hollerith constants.
- Mixed-mode expressions — can contain any data type, including complex and byte.
- End-of-line comments — any FORTRAN statement can be followed on the same line by a comment that begins with an exclamation point.
- READ/WRITE end-of-file or error condition transfer — END = *n* and ERR = *n* (where *n* is a statement number) can be included in any READ or WRITE statement to transfer control to the specified statement upon detection of an end-of-file or error condition. The ERR = *n* option is also permitted in the ENCODE and DECODE statements, allowing program control of data format errors.
- General expression DO and GO TO parameters — general expressions are permitted for the initial value, increment, and limit parameters in the DO statement and as the control parameter in the computed GO TO statement.
- DO increment parameter — the value can be negative.
- Optional statement label list — in an assigned GO TO.
- Default FORMAT widths — input or output formatting can be specified by type and default width, and precision values will be supplied.
- Additional I/O statements — these include file control and attribute definitions, list-directed (free-format) I/O, device-oriented I/O, memory-to-memory formatting, and unformatted direct-access I/O to read and write files written in any format.
- Logical operations on INTEGER data — the .AND., .OR., .NOT., .XOR., and .EQV. logical operators can be applied to integer data for bit masking and manipulation.

Optimization

The FORTRAN-20 compiler uses optimization to produce an object program that gives the same results as the original unoptimized program, but takes significantly less execution time.

Optimization techniques include:

- Eliminating redundant computations
- Substituting arithmetic operations with operations that take less time
- Removing constant computations from DO loops
- Folding and propagating constants
- Removing inaccessible code
- Allocating global registers
- Optimizing I/O
- Detecting uninitialized variables

Debugging Tools

Three debugging facilities are available to the FORTRAN programmer: FORDDT debugger, the FORTRAN object-time system, and the use of a "D" in Column 1 of a FORTRAN statement.

Using FORDDT, a programmer can set up to 10 runtime breakpoints on subroutine entries or on source statement labels or line numbers. The programmer can also continue from a breakpoint, display and change the value of variables, locate symbols, execute source lines, and display a trace of subroutine calls.

The debugger commands are similar to FORTRAN statements, so that programmers can easily learn and use them.

With the exception of a line-by-line trace or the inclusion of PAUSE statements, none of the FORDDT debugging features increases the execution time of a program. FORDDT also provides an entry into the system debugger, DDT.

The FORTRAN object-time system (FOROTS) provides a traceback feature that locates and lists the actual program unit and line number of a fatal runtime error. If the program unit is a subroutine or function subprogram, the error handler traces back to the calling program unit and displays the name of that program unit and the line number where the call occurred. This process continues until the calling sequence is traced back to a specific line number in the main program. The traceback feature helps the programmer to determine the exact location of an error, even if it occurs in a deeply nested subroutine.

A "D" in Column 1 of a FORTRAN statement allows that statement to be conditionally compiled. These statements are considered comment lines by the compiler unless the appropriate debugging line switch is issued in the compiler command string.

Liberal use of the PAUSE statement and selective variable printing provide programmers with a method of monitoring program execution. This allows the inclusion of debugging aids that can be compiled in the early program testing stages and later eliminated without source program modification.

COBOL

COBOL-68 and COBOL-74 are optional high-level languages designed specifically for business data processing. COBOL can be used to create online terminal applications or to write batch applications. Under control of the multistream batch processor or from an interactive terminal, program and data decks can be loaded into the cardreader for spooling operations. Both methods use the same command language.

COBOL features include:

- ANSI-standard compliance
- Quick, efficient program development
- Simple, interactive user application interface
- Efficient operation in both batch and online operation
- Programming tools with powerful, easy-to-use data editing, sorting, updating, and reporting features



- Remote access from online terminals or from remote stations
- Complete program and device independence for efficiency and reliability
- Ability to call subroutines written in COBOL, FORTRAN, or MACRO

Listings produced by the compiler contain many documentation and debugging aids. English diagnostic messages are embedded in the source listing at the point of error. The listing can also include, at the user's discretion, a complete map of the object program and an easily read listing of the compiled code similar to that used by the MACRO assembler. All object code is expanded to list the machine mnemonics and user-defined names in addition to the binary machine code (in octal). There is also an implementation of the COBOL REPORT WRITER statement.

Data Types

COBOL supports all of the common COBOL data types, including:

- Numeric COMP-3 packed decimal
- Numeric COMPUTATIONAL (COMP) binary
- Numeric COMPUTATIONAL-1 (COMP-1) floating point
- Alphanumeric DISPLAY (ASCII, EBCDIC, or SIXBIT)
- Numeric DISPLAY (ASCII, EBCDIC, or SIXBIT)

These data types are required in a variety of applications and are provided for flexibility in specification and design.

String Manipulation

COBOL provides INSPECT, STRING, and UNSTRING verbs for handling character strings. With these verbs programmers can search for embedded character strings with TALLY and REPLACE. The verbs can join together or break out separate strings with various delimiters.

Interactive COBOL Execution

The ACCEPT and DISPLAY statements of the PROCEDURE DIVISION enable easy, terminal-oriented interaction between a COBOL program and the program user.

The program can receive user input lines with the ACCEPT statement. The ACCEPT statement also can retrieve the current date or time from the system.

The DISPLAY statement transfers data from a specified literal or data item to a specified device, normally the user's terminal. The DISPLAY WITH NO ADVANCING statement inhibits the automatic appending of a carriage return and line feed to halt the device after the last character is displayed. This is especially useful when typing prompting messages on the terminal.

File Organization

The sequential I/O, relative I/O, and indexed I/O modules meet the ANSI-74 high-level standards, except as indicated in Table 8-1, and include all the COBOL verbs.

The COBOL object-time system includes a complete indexed sequential access mode (ISAM) package to enable the user to access data either sequentially or randomly by key value on random-access devices. The time required to access a file is minimally affected by the number of additions made to the file. The technique of "chaining" records is not used. Instead, the index to the file is updated to minimize the number of accesses necessary to retrieve records.

Library Facility

With COBOL, programmers have a full ANSI-74 high-level library facility that includes high-level extensions (COPY...REPLACING). Frequently used data descriptions and program text sections can be held in library files that are available to all programs. These files can be copied at compile time to reduce program preparation time and to eliminate a common source of errors.

ENTER Facility

The ENTER statement allows COBOL programs to invoke separately compiled subprograms, passing arguments in the process. These subprograms can be written in COBOL, FORTRAN, or MACRO. The ENTER facility:

- Provides flexibility through modular development of application systems
- Permits functional separation of small, well defined source modules
- Gives the programmer access to operating-system-dependent features via subroutines written in MACRO

Online Debugger

The online COBDDT debugging package permits user interaction during the execution of a program. Source programs require no modifications to use COBDDT — it is simply loaded with the object program when execution starts. The user can specify points within the program at which to pause during execution, so that the user can examine and modify the contents of data items before proceeding. All references to data and procedure items are made by using the name in the source program. The user talks to the debugging package with familiar names rather than truncated or substituted names.

Source Program Input

The disk-resident compiler can accept source program

input from all supported input devices, including source text library files stored on disks.

COBOL accepts source programs that are coded using either the conventional 80-column card reference format or the short, easy-to-enter terminal format.

Terminal Format is used with context editors controlled from an online terminal keyboard. It eliminates the line number and identification fields and allows horizontal tab characters and short lines. These capabilities offer potential savings in disk space and allow easier interactive input of source programs.

Conventional Format produces source programs that are compatible with the reference format of other COBOL compilers throughout the industry.

RERUN

RERUN allows the user to periodically save the status of a job. In the event of a later disruption the job can be restarted from the point of the last status saved instead of from the beginning.

COBOL-68 and COBOL-74

COBOL on DECSYSTEM-20s is available in two versions: COBOL-68 and COBOL-74.

COBOL-68 is an implementation of the COBOL language based on the ANSI COBOL X3.23-1968 standard. COBOL-74 is an implementation of the COBOL language based on the ANSI COBOL X3.23-1974 standard. COBOL-74 meets the high-level requirements of Federal Information Processing Society (FIPS) PUB 21-1 as tested by the FCCTS (Federal COBOL Compiler Testing Service) with the exceptions noted in Table 8-1.

Table 8-1 COBOL-74 Support Levels

ANS-74 Module	Level Supported by COBOL-74	FIPS Pub 21-1 "High Level" Requirements
Nucleus	2	2
Table Handling	2	2
Sequential I/O	2	2
Relative I/O	2	2
Indexed I/O	2*	2
Segmentation	2	2
Library	2	2
Debug	2†	2
Interprogram Communication	2	2
SORT/MERGE	2	2
Communication	‡	2
Report Writer	§	2

* Alternate key not supported

† Debug module is functionally replaced by COBDDT

‡ Not supported

§ ANSI-66 Report Writer syntax plus SUPPRESS statement

ALGOL-20

ALGOL-20 is a scientific language designed for describing computation processing or algorithms. It is a problem-solving language in which the problem is expressed as complete and precise statements of a procedure.

The TOPS-20 interactive editors and an integrated ALGOL debugger make it easy to code, test, and debug programs. Program modularity is available through block structure, subprograms, and separately compiled procedures.

ALGOL-20 is an implementation of the ALGOL-60 language. The compiler consists of a 14K-word reentrant (sharable) segment and a data segment that varies in size depending on the size of the program to be compiled. The one-pass, single-phase compiler produces diagnostics and generates optimized object code.

ALGOL-20 language features include:

- Long reel scalars, arrays, and procedures using the double-precision hardware with 62-bit mantissas
- String scalars, arrays, procedures, and byte manipulation for users to generate, manipulate, and input or output strings or individual bytes ranging in size from one to 36 bits
- Assignments within expressions
- A remainder operator
- Unique implementation of dynamic arrays
- Octal Boolean constants and integer-to-Boolean and Boolean-to-integer conversion functions
- WHILE statement and a convenient abbreviated form of the FOR statement
- The ability to call FORTRAN language subroutines and functions
- Identifiers up to 64 characters long

Block Structure

The ALGOL program structure is somewhat more sophisticated than some other high-level languages such as FORTRAN or BASIC. ALGOL programmers can effectively use structured programming techniques. An ALGOL program has a number of hierarchically arranged blocks consisting of declarations and statements enclosed by the words BEGIN and END. Scope rules and declarations determine which variables are global and which are local to a block.

Block structure offers many advantages including easier implementation of top-down application design, code modularity, easily read and understood programs, and increased protection from side effects.

Procedures

ALGOL procedures are subprograms that are useful in writing highly structured, easily read programs. Code segments that are used more than once, and/or that can be viewed as a module of the larger program, can be separated into procedures. To aid top-down program development, procedures can be written, debugged, and compiled separately from the main program. Large structured programs frequently consist of a series of procedure calls.

Parameters can be passed to a procedure by *value* or by *name*. When an expression in a procedure call (an actual parameter or argument) is passed by value, a copy of its value is made available as a local, formal parameter (dummy variable) within the procedure. This is an efficient way to pass many expressions and to protect the calling block from side effects.

When an argument is passed to a procedure by name, any changes made to the value of the formal parameter in the procedure also changes the actual parameter as if it, instead of the dummy, appeared in the procedure body. Passing an array or a string argument by name instead of value saves memory space and allows the procedure to treat the argument as a global variable.

ALGOL supports recursive procedures that can call themselves directly or indirectly to a depth limited only by the user's available memory space.

Compiler and System Features

The ALGOL compiler reports all source program errors on the user's terminal or on a listing device. The ALGOL-20 compiler adds these extensions to ALGOL-60:

- Long real data type equivalent to FORTRAN's double-precision data type for more accurate real computations
- External procedure can be compiled independently of main program
- Convenience in loop implementation with WHILE and abbreviated FOR statements
- Programmer can manipulate strings of various-sized bytes and can individually manipulate the bytes within a string through byte subscripting
- Integer remainder function
- Delimiter words can be represented in either reserved word format or as nonreserved words
- Constants of data type real can be expressed as an integer part or as a decimal part only

OWN Variables

Special integer, real, long real, Boolean, and string variables called OWN variables have these properties:

- They follow the normal scope rules within a block
- When control passes outside the block, the values are retained and still available when the block is reentered
- Their values are initialized to zero, false, or null

OWN variables are valuable in large-array work inside procedures because the array doesn't have to be initialized every time it is referred to. Arrays defined and used inside procedures are comparable to subroutines in other languages, but they can be dynamically defined and redefined.

Switches

Switches provide the function of the CASE statement — program control jumps to various locations depending on the value of an expression. Switches automatically detect when the evaluated expression is out of range.

String Constants

String constants enable the user to refer to strings of ASCII characters within a program by using a variable name. The

length of the string is limited only by available memory. String constants are typically used to convey messages to the program users or to assign value to string variables.

Object-Time System

The ALGOL-20 object-time system (ALGOTS) provides a basic input/output system so that the user can communicate with directoried and nondirectoried devices in ASCII and binary modes.

At run time, ALGOTS provides I/O processing, storage management, and debugging facilities. ALGOTS consists of a sharable high segment of 11K words and a low segment of at least 2K words for each program depending on object program size, library routines used, and heap and stack sizes. The object system provides a checking mode for selective testing of array subscript bounds.

The object-time system includes a library of routines that can be incorporated into a user's program, including:

- A set of mathematical functions of both single- and double-precision
- Maxima and minima functions
- String manipulation routines
- Bit field manipulation routines
- FORTRAN subprogram interface routines

The runtime facilities include:

- I/O with directory and nondirectory devices in both ASCII and binary modes. Up to sixteen internal logic channels plus default terminal I/O channels are available.
- Storage management of the heap and stack enables the program to borrow a temporary buffer for I/O, OWN arrays, and dynamically created byte strings. It also provides memory expansion when needed.
- The ALGOL dynamic debugger allows interruption of program execution, setting and clearing of breakpoints, examination and alteration of ALGOL variables that are in scope, examination of various system parameters, automatic typing of ALGOL variables after a breakpoint, examination of the code generated, and continued program execution both from a halt and from an appropriate label.

BASIC-PLUS-2

Although BASIC-PLUS-2 is an ideal language for novice programmers who need a fast, easy way to solve problems, it is also a powerful, efficient language suitable for sophisticated applications.

BASIC-PLUS-2 retains Dartmouth BASIC's simplicity and ease of use. It adds many features and greatly enhances performance to accommodate a wide range of applications. The BASIC-PLUS-2 compiler produces fast-running programs, yet its immediate mode retains the advantages of the much slower BASIC interpreters.

BASIC-PLUS-2 uses features of the TOPS-20 operating system to provide immediate mode and compile-and-go execution. The user can also save compiled processes as directly executable modules. BASIC-PLUS-2 features include:

- Immediate "desk calculator"/debug mode
- Ability to save compilations
- Error diagnostics and program-controlled error-handling routines
- Subprogram capabilities to help structure programs
- IF-THEN-ELSE, UNLESS, WHILE, and UNTIL eliminate most of the need for GOTO statements
- Block formatting with unlimited statement length, multi-line statements, and multistatement lines
- Variable names of up to 30 characters make programs more readable and easier to document
- Full matrix manipulation and arithmetic
- Powerful string-handling functions
- Arrays stored in files, yet can be accessed the same as in-memory arrays
- Other data file structures, including ASCII text files and sequential, random, and multikey indexed record files
- Formatting facilities for output and reports

File Capability

BASIC-PLUS-2 gives users high-speed, online access to disk files. Files can be created, updated, extended, and deleted under program control.

Any number of programs can simultaneously read data from the same file, but only one write operation can occur at a time. Multiple updates are handled safely through the UPDATE feature. UPDATE locks a disk record from other programs while one program is updating it, so all programs access current, valid files. Programs can be written to recover cleanly from failed attempts to open or access files.

Virtual Memory Arrays

Many applications need to address and update random records on a disk file. Other applications might require more memory for data storage than is economically feasible. BASIC-PLUS-2 fills both of these requirements with *virtual arrays*, a simple, random access filetype.

With the virtual memory array facility, programmers can specify that a particular data matrix is not to be stored in the computer memory, but on disk. Data, stored on disks that are external to the user program, can be retrieved by name at a later session. Items within the file are individually addressable.

Using the virtual memory array facility, programs can operate on data structures that are too large to be wholly in memory. The disk file system is used for storing data arrays, and only portions of these files are maintained in main memory at any given time.

With virtual data storage, programs can reference any element of one or more matrices within the files, no matter where in the file the element resides. Random access allows programmers nonsequential referencing of the data for use in any BASIC statement. Virtual memory matrix elements are read into memory automatically by the system.

Virtual memory arrays are stored as unformatted binary data. This means that no I/O conversion is performed when storing or retrieving elements in these arrays.

Record Input/Output

Three methods for performing record I/O operations are provided: formatted ASCII I/O, I/O to virtual memory arrays, and block I/O.

Record I/O permits the user program to have complete control of I/O operations. Properly used, it is the most flexible and efficient technique of data transfer available under BASIC-PLUS-2.

Language extensions efficiently handle records composed of fixed-length fields. Block I/O offers great flexibility and performance for input and output device transfers. For example:

- GET and PUT statements initiate I/O operations for disk and magnetic tape, and read and write sequential data blocks.
- For relative disk files, the programmer can have complete random access to the file by specifying a record number in the GET, PUT, or FIND statement.
- For indexed disk files, the programmer can access records by multiple key values.
- Programmers use MAP and MOVE statements to map an I/O buffer. The MAP statement defines the format of a record when that format can be specified at compile time. If the format cannot be specified at compile time, the MOVE statement can be used to dynamically access the data in a record.
- A set of numeric/string conversion functions permits optimum packing of number fields, thereby saving space in the record.
- Programmers use formatted or stream ASCII I/O to handle character strings of indeterminate length. System facilities handle the blocking and unblocking of formatted ASCII files.

Data Formats and Operations

Programmers can manipulate string, integer numeric, or floating-point numeric data.

String variables are sequences of ASCII characters that are treated as units. Programmers can define string constants and string variables, including subscripted variables. In addition, relational operators can be applied to string operands to compare and indicate alphabetic (ASCII) sequence. Using the CHANGE statement, individual string characters can be converted to their equivalent ASCII code in decimal, and vice versa.

A variety of string functions enables programmers to concatenate two strings, access part of a string, determine the number of characters in a string, search for substrings, convert strings to compact storage formats, and define new string functions. Character strings can be any length, limited only by available memory.

String arrays further enhance the usefulness of string variables. An entire list of alphabetic data can be input, processed, and output with only two statements.

Usually, all variable and constant numeric values specified in a BASIC-PLUS-2 program are stored internally as floating-point numbers. If operations such as counting, indexing, and subscripting deal with integer numbers, sig-

nificant economies in storage space can be achieved by using the integer data type. The integer data type requires only one word of storage per value. Integer arithmetic is significantly faster than floating-point arithmetic.

Single- or double-precision, floating-point numeric operations are the default numeric type. Single-precision floating-point numbers occupy one word of storage in memory, and double-precision numbers occupy two.

Programmers working with floating-point numbers can increase accuracy of operations involving fractional numbers by using the string arithmetic functions or the scaled arithmetic feature.

Programmers can perform arithmetic operations using a mix of integer and floating-point numbers. If both operands of an arithmetic operation are either explicitly integer or explicitly floating-point, the system automatically generates integer or floating-point results. If one operand is an integer and another is floating-point, the system converts the integer to a floating-point representation and generates a floating-point result. If one operand is an integer and the other operand is a constant that can be interpreted either as a floating-point number or an integer, the system generates an integer result. Programmers can explicitly impose the formats, thereby controlling the result of the operation.

Matrix Manipulation

Arrays are indexed variables containing data of any one type — integer, floating-point, or string. Arrays can have one or two dimensions, and the number of elements in each dimension is implicitly or explicitly declared by the programmer. The number and size of an array's dimensions are explicitly defined with DIM, MAP, or COMMON statements. They are implicitly defined by declaring a subscripted variable. The size of an array is limited only by available memory.

Programmers can alter the number of elements in each row and the number of columns in the matrix and can input, print, add, multiply, transpose, invert, or take the determinant of entire data matrices in a single operation.

Conditional Statements and Program Segmentation

BASIC-PLUS-2 extends the BASIC language by including several additional statements for easier logic flow and function definitions. The IF-THEN-ELSE, WHILE, UNTIL, GOSUB, function definition, and CALL statements provide a variety of loop controls, as well as function and procedure subprogram capabilities. The programmer can use multiline statements, multistatement lines, and the subroutine features to write structured programs, without GOTO statements.

The CALL statement accesses external subprograms. Programs can comprise several modular segments, each of which can be compiled separately to speed program development.

The CHAIN statement transfers control from the current program to a program stored in a file. CHAIN loads a program from the file, compiles it if necessary, and starts its execution from the beginning or from any line number.

Subprograms or programs connected by a CHAIN share data through COMMON statements.

The ON ERROR GOTO statement allows programmers to write subroutines to handle error conditions normally considered fatal. The program can test a system variable to determine which error occurred, and can examine another system variable to determine the line number at which the error occurred. The action taken depends on the error and, if desired, on the line number of the error. For example, a program user might specify a nonexistent data file for input to the program. The error-handling routine could print a message such as "no such file, try again," and then return execution to the line that requests the data file name.

The SLEEP and WAIT statements allow program suspension, either for a specified time interval or until input from a terminal is received.

Debugging Tools

Dynamic debugging tools maximize program debugging efficiency by minimizing the need to rerun programs. Programs can be interrupted at any point to check, correct, and then resume operation. The most useful of these tools are immediate mode statement execution, the STOP statement, and the CONTINUE command.

Using immediate mode, programmers enter a statement without assigning a program line number, and then immediately execute that statement. A programmed and an immediate mode statement are distinguished solely by the presence or absence of a line number.

Immediate mode is also useful for performing simple calculations that occur too infrequently to warrant being programmed.

When debugging, programmers can use STOP statements to create executable sections of their program. Upon execution, a STOP statement halts the program and displays a message indicating where the program was halted. The programmer can then use immediate mode to examine and/or change data values, or to add, to delete, or to modify lines. CONTINUE or GOTO statements are used to continue program execution.

APL

Two levels of this popular language are available, and both are full APL implementations with significant extensions. The basic version suits users who do not require the file I/O or the advanced APL functions. The extended version of APL, APL-SF, has all of the features of the basic version plus advanced features that substantially increase its useful range of applications. Both feature:

- Full implementation with significant extensions
- Fast execution
- System functions/variables
- User-definable error trapping
- Error analysis and recovery
- Support of highly interactive applications
- Double-precision arithmetic

- Integrated debugging features
- Workspace interchange features or capabilities

APL-SF enables the user to obtain canonical string representations, to create local functions, to erase and classify names from a workspace, and to perform various file I/O operations including ENQ/DEQ. APL-SF features:

- System variables with which the programmer can set tolerances, index origins, and store accounting information
- File I/O that makes it easy to structure program data and interchange data files between other DECSYSTEM-20 languages such as FORTRAN and COBOL
- System commands with which the programmer can redirect terminal input and output to any file
- A function that can solve linear equations, take the inverse of a matrix, or solve an overdetermined set of linear equations using a least-squares fit
- A function that converts numeric data to a character string and enables the programmer to write user-defined functions that perform special output formatting and function editing
- A function that efficiently finds the indices in a vector for which a particular Boolean expression is true
- A function that executes a character as an APL statement
- A convenient and efficient mechanism for formatting output data; for example, an entire table with associated text can be formatted in a single operation, or a large matrix can be formatted with alphanumerics

APL uses one of the most concise, consistent, and powerful character sets ever devised. APL is especially suited for handling array-structured alphanumeric data. It is also used as a general data processing language and a mathematical tool.

APL allows programmer-defined functions and primitive language functions to be expressed with the same syntax. Thus, programmers can expand the capabilities of the language to handle the requirements of any application.

Data Structures

APL supports a variety of numeric and character data structures. They are:

- Scalars, which are a single numeric or character value with no dimensions
- Vectors or lists, which are one-dimensional arrays or character strings consisting of any number of values
- Matrices or tables, which are two-dimensional arrays consisting of rows and columns
- Arrays with three or more dimensions — the maximum number of dimensions and the maximum size of an array are limited only by the size of the workspace

Interacting with APL

Programmers interact with APL using a hardcopy or video terminal. DIGITAL's LA37 and LA120 hardcopy terminals include an APL/ASCII dual-character set. On other terminals, keyboard mnemonics can represent special APL characters.

System Commands, Functions, and Variables

Programmers can change system parameters, determine hardware or operational characteristics, and modify workspace parameters through system commands, system functions, and system variables. System commands control the operational environment in which an APL session is conducted by allowing programmers to examine or change the state of the system. System functions and variables are used to communicate with the APL system to change user workspace characteristics and to report statistics about the workspace and the APL system.

Statements

There are two types of statements in APL programs: assignment and branch. Assignment statements include calculation and input/output operations. Branch statements are used to restart a function or to handle the transfer of control from one part of a program to another. Branch statements are relevant only to programmer-defined functions.

An APL statement can contain:

- Identifiers
- Constants
- APL primitive functions
- User-defined functions

APL Statement Execution

APL language statements operate in either of two modes:

- *Immediate or execution mode* — in this desk-calculator mode, APL statements and expressions entered by the user are executed immediately
- *Function-definition mode* — in this mode, APL programs and functions are developed, edited, named, and saved for later use

Programmers can shift from one mode to the other. The syntax of the APL language is identical in both modes.



Debugging Tools

Function execution is suspended if an error occurs or if a stop vector is set. When execution is suspended, the program displays the name of the suspended function and the line number of the statement that would have been executed next or that was being executed. APL then awaits input in immediate mode. Programmers can perform any other APL operations at this time. The programmer can resume execution after fixing the problem and can observe function nesting.

Programs can also automatically display intermediate results of function execution. As a program-tracing aid, the values computed by one or more function statements can be output each time those statements are executed.

Function execution can be suspended from within the function itself. A stop control vector with a syntax similar to that of the trace vector suspends function execution just before execution of one or more specified statements.

Workspaces

An APL workspace is a buffer in the programmer's memory area that stores the functions, variables, values, and temporary results obtained while executing APL statements. Using APL system commands, workspaces can be saved, loaded, and erased. Workspaces are stored as DECSYSTEM-20 disk files and can be manipulated as such.

Through the file I/O facilities of APL-SF, a user can convert an APL workspace to ASCII format, and then edit it with any DIGITAL-supplied editor.

File Organization

The APL-SF file system allows access to data and program files on a variety of system devices. The file system, implemented as an integral part of the APL language, provides an interface to the TOPS-20 operating system.

The APL file system support is provided by system functions for assigning, closing, synchronizing, and renaming files, and by file functions for input and output operations on ASCII data, APL data objects, or arbitrary binary data in either sequential- or random-access modes.

APL supports ASCII sequential- and random-access files. ASCII sequential data files can be read and written sequentially by any TOPS-20 language processor (e.g., BASIC-PLUS-2, FORTRAN, MACRO). APL can treat random access files as random access memory. Programmers can access any word in the file directly by specifying the individual word or value to be read or written. APL arrays of any shape or size can be read from or written to APL files either sequentially or randomly by component (i.e., record) number.

Error Analysis and Recovery

Instead of stopping execution, APL-SF error analysis and recovery permit the program to take remedial action.

There are two ways to trap errors:

- The programmer can execute an expression suspected of causing an error, then test whether an error occurred.
- The programmer can specify that whenever an error oc-

curs, a particular APL expression be executed. Within that expression, the error can be analyzed and remedial action taken.

Error-trapping features enable computer-assisted instruction applications. A programmer can, for example, write a program that lets a student write a problem solution. The controlling program can intercept any student error.

Conversion Package

A complete package is available to convert non-DIGITAL APL workspaces and data files into APL-SF workspaces and data files. The package code is mostly APL and is easily adaptable for use with other APL systems. It is written according to the workspace interchange convention adopted by the SIGPLAN Technical Committee on APL (STAPL) of the Association for Computing Machinery (ACM).

BLISS-36

BLISS-36 is DIGITAL's implementation language for software development. BLISS is an optimizing, high-level systems-implementation language for the DECSYSTEM-20. It is designed for building compilers, utilities, and operating system software, and encourages the writing of highly structured, easily maintained programs.

BLISS has the features of a modern, structured high-level language combined with the flexibility of an assembly language. BLISS can help systems programmers be more productive, shorten project development time, and lower maintenance cost.

The key features of BLISS are:

- State-of-the-art optimization technique to generate highly optimized programs
- A full set of structured programming constructs including IF-THEN-ELSE, CASE, DO-WHILE, SELECT, and DECR statements
- Sophisticated macro processing capabilities
- Access to machine-dependent features including PSECTS, hardware registers, and machine instructions (including UUO and JSYS)
- A linkage declaration that supports user-selected register conventions. Users can rebuild the object-time system to support nonstandard register conventions
- Precompiled source libraries similar to MONSYM

Compiling

The compiler can display a listing of errors and warning flags on the programmer's terminal or can generate a listing with the errors and flags embedded. Listings can be printed as needed, and most syntax errors are labelled as such.

Debugging

An interactive symbolic debugger supporting BLISS-style expression evaluation is used.

File Organization

BLISS can access any file organization.

Compatibility with Other Languages

BLISS is not intended to replace the other high-level languages such as COBOL, FORTRAN, or BASIC; instead, it complements them. Programs written in other languages, except BASIC-PLUS-2, can call BLISS routines through the standard operating system calling sequence. BLISS programs can call routines written in a variety of other languages, and they use several linkage conventions.

CPL

Conversational Programming Language (CPL) is a PL/I subset interpreter. It is well documented, easy to learn, and a subset of ANSI-1976 PL/I.

At the user's option, statements are executed immediately or saved for deferred execution. A beginning programmer can start by executing simple computational statements and proceed to building programs.

Since CPL is an interpreter, a user can track programs very closely. Debugging features include source-level breakpoints and program modification.

CPL includes:

- ANSI PL/I statements — ALLOCATE, ASSIGNMENT, BEGIN, CALL, CLOSE, DECLARE, DEFAULT, DELETE, DO, END, FORMAT, FREE, GET, GOTO, IF, NULL, ON, OPEN, PROCEDURE, PUT, READ, RETURN, REVERT, SIGNAL, STOP, and WRITE
- Data types — FIXED, FLOAT, CHARACTER, CHARACTER-VARYING, BIT, BIT-VARYING, POINTER, and arrays of these types
- Storage classes — AUTOMATIC, STATIC, CONTROLLED, and BASED
- Recursive procedure support
- Almost all ANSI PL/I arithmetic, mathematical, string-handling, array, and storage control built-in functions
- STRING, SUBSTR, and UNSPEC pseudovariables

Immediate Mode

The CPL immediate or desk-calculator mode is integrated

into the system so that the user can easily move between programming and calculating.

Almost any CPL PL/I language statement can be typed without a line number for immediate mode execution. A user can perform simple computations to get immediate answers, and can assign intermediate results to variables and use the variables in further computations. Immediate mode is thus useful for debugging.

Program Creation

Programmers can create CPL programs without using an editor or utility. Program statements are typed to CPL; insertions, changes, or deletions are done easily.

CPL provides automatic, immediate syntax checking of statements. If there is an error, CPL tells where the error occurred. The user then can correct the statement and continue. This avoids waiting for a compilation to provide a list of errors.

Programs created using an editor or file manipulation facility are checked automatically when loaded into CPL. Any erroneous statement will be listed with an error message that indicates the problem.

Debugging

CPL is the only PL/I language processor available that provides true source-level program debugging. To debug a program, a programmer can:

- Load the program.
- Set breakpoints on any statement.
- Execute the program.
- Use CPL statements in immediate mode to isolate problems when the program stops on an error, a breakpoint, or at programmer command. Variables can be examined or changed. I/O can be done. New variables can be created to store intermediate results.
- Make source modifications to the program, such as add, delete, or modify statements, and set or clear breakpoints.
- Continue execution after the current halt in execution.

9 Data Management and Application Products



In addition to the standard file management facilities described in Section 4, TOPS-20 has optional data management and application products.

DBMS is a CODASYL-compliant database management system with which users can organize and maintain data in customized databases, and rapidly and conveniently access that data. IQL is an interactive query language for information retrieval and report writing. IQL can access DBMS files. SORT/MERGE is a TOPS-20 sort utility that operates by itself or with COBOL or FORTRAN programs to reorder the records of files into new sequences or to merge sorted files into a single, sorted file. TRAFFIC-20 is a collection of CRT screen-formatting and program-to-program communications subroutines useful in writing online COBOL transaction processing applications.

Two application products, COGO-20 and PCS-20, are also available. COGO-20 includes a geometric language for solving problems in plane coordinate geometry. PCS-20 is a project control system that analyzes critical path or procedure networks and generates a number of resource, cost, and critical path reports.

The data management products detailed in this section are:

- DBMS — a CODASYL database management system
- IQL — an interactive query language for information retrieval and report writing
- SORT/MERGE — a sorting and merging utility for TOPS-20 files
- TRAFFIC-20 — a collection of screen-formatting and program-to-program communications subroutines for COBOL programs

The optional application products discussed are:

- COGO-20 — a tool for solving problems in plane coordinate geometry using a geometric language
- PCS-20 — a project control system that analyzes critical path or procedure networks

The comprehensive data management system that is part of TOPS-20 and included with all DECSYSTEM-20s is discussed in Section 4 of this technical summary under the headline *The File System*. Access to data files from various programming languages is described in Section 8.

DBMS

DBMS is an optional TOPS-20 CODASYL-compliant database management system that lets users organize and maintain data in customized databases and provides rapid and convenient access to the data.

DBMS integrates related processes and data structures. It is used when traditional file management techniques would be difficult, costly, inadequate, and/or prone to error. DBMS features:

- COBOL and FORTRAN interfaces
- English-language interface for inquiry and report generation through the IQL package discussed below
- Easy SCHEMA/SUBSCHEMA definition and update
- Full journaling and database recovery capability
- Protection against unauthorized database access with centralized control of privacy
- High throughput
- Temporary work areas for use during development to protect database integrity

CODASYL Compliance

DBMS provides software to define, access, and maintain data in the network structures of an integrated database. DBMS is based on the CODASYL Database Task Group Report of April 1971.

Data Description Process

Data structures can be established either in a hierarchical form (a *tree*) or in networks with multilevel relationships. Relationships can exist within or between files with no fixed limit on the length of chains. The TOPS-20 file system is used to construct data base areas. The number of relationships involving any record and the size of the record is normally limited only by the design constraints of the application.

Data Manipulation Process

Database records can be referred to via data manipulation statements included in COBOL or FORTRAN application programs. These statements are used to store, modify, and delete fields and records that are of specific interest to an authorized application user. Records can be inserted and removed within structural relationships, and can be directly accessed by symbolic keys or by movement through structural relationships.

DBMS Modules

TOPS-20 DBMS modules include the Data Definition Language, Data Manipulation Language, and Database Control System.

The *Data Definition Language (DDL)* equips the database administrator to centrally define the logical and physical characteristics of the database files and database records. By controlling the placement of records on the physical storage medium, the administrator can assure both data security and high-speed data retrieval.

Records can be grouped logically in "sets", which are used to model the real-world relationships among objects. Instead of storing extra data about relationships, DBMS-20 links related records in logical chains.

This thrust toward effective real-world modeling is carried a step further in transaction definitions. A transaction definition is an integrated series of manipulations performed against the database. A transaction defines the context of these interactions, allowing the user to perform a series of data manipulations as a single, integrated operation.

The database definitions reside in a central location in the SCHEMA, and can be accessed by user programs through the SUBSCHEMA. A SUBSCHEMA is a subset of the SCHEMA that is defined by the administrator to satisfy the needs of related groups of programs. By restricting the scope of the SUBSCHEMA, the administrator can prevent certain classes of users from accessing portions of the database, from the level of files down to the level of data items. This feature provides one of the inherent security checks of DBMS-20.

The *Data Manipulation Language (DML)* performs the database retrieval and update. The programmer includes DML statements in a COBOL or FORTRAN program. COBOL and FORTRAN are called the host languages for DML.

The first DML statement in a program is the INVOKE statement, which provides the program with the definitions from the SUBSCHEMA, in a form suitable for the host compiler. No further data declarations are needed because the centrally defined database structures are mapped directly into the host program.

DBMS-20 provides multithreaded simultaneous update. Many users can access the database at the same time. Each program is protected by a system of locks that are controlled in the Schema. The database administrator can grant exclusive control to one program, or distribute it among many programs. When a large number of programs are using the database, each one locks the resources it is using until it completes its transaction. The

protective locks can be placed on areas (files) being used, or on the specific database page.

Two typical DML verbs are FIND and STORE. In DBMS-20, there are six ways to find a record: by traversing a set, by hashing a data item, by sorted order, by direct address, by relative address, or by current program context.

When a record is read from disk, DBMS-20 can also read in other database pages containing related records. This operation, called clustering, requires only a single disk access for the entire page cluster.

The STORE statement, in addition to placing the record on the database, automatically performs a number of operations. In accordance with the specifications of the Schema, a record can be stored next to its logical group, or can be distributed randomly across the database. The Schema can specify how many records to place on a single page, or how many pages to leave between records. The STORE operation can automatically choose an appropriate set and link the record to it immediately.

An important, but largely invisible component of DBMS-20, is the journal, which is a record of everything that has happened to the database. When using the journal, a program can recover from any kind of error that affects the database — either automatically or under program control, the database is restored to the state it was before the current transaction began. The journal helps assure the consistency, integrity, and reliability of the database.

Database Control System (DBCS) that is a runtime interface between a COBOL or FORTRAN program and the TOPS-20 operating system. The DBCS exists in a specific module within the COBOL object-time system (LIBOL) and the FORTRAN object-time system (FOROTS). Failed DML updates are recovered via the journaling facility.

DBMS Utilities

The DBMS utilities help the database manager run and maintain the database. The utilities include:

- **DBMEND** — for repairing the database using the journal. DBMEND can restore the database after a system software or hardware failure, or a user error.
- **DBINFO** — to generate a variety reports about the structure and contents of the database, including maps and dictionaries from the Schema. Users can also obtain data dumps of single pages, single sets, single records, or any larger portion of the database.
- **STATS** — to generate complete statistics about the performance of the database. These statistics, relating to buffer management, disk I/O, lock performance, and DML runtime, allow the database administrator to evaluate the database design and fine-tune it for maximum performance. STATS can be called by a database program.

IQL

The optional Interactive Query Language (IQL) is used with TOPS-20 for information retrieval and report writing. It takes requests written in English-like formats, reads one or more input files, and processes the data according to the request. IQL queries are groups of easily written, re-

port-generating commands. IQL can access TOPS-20 files and interface to DBMS to provide a powerful, fully integrated database management system.

IQL features:

- Multiple input files (DBMS, ISAM, sequential)
- Extensive record selection
- Sorting
- Conditional processing — AND/OR conditions can be strung together; parentheses can be nested nine deep
- Built-in summary statements
- Complete report-formatting capabilities including multiple across labels and special forms
- Multiple reports — up to nine, standard, and expandable
- Files can be output in original or new format
- Matrix reporting by manipulation of summaries or individual items
- Powerful computation capabilities
- Built-in summary statements for tallies, totals, and averages
- File-updating capability
- Dictionary presorting of file, record, and item information, including printing of column titles and pictures
- Default automatic formatting of reports, including field alignment, dates, paging, and columns
- Interactive or batch modes of operation
- Exits to user-written modules

Traditionally, data manipulation and reporting has been left to programs written in COBOL, FORTRAN, or assembly language. This method incurs high programming costs and delays in obtaining report information. IQL, however, is an inquiry system that can quickly extract, summarize, reorganize, report, and copy file information. Table 9-1 illustrates the use of IQL Ad Hoc inquiry and reporting.

Interactive Mode

In interactive mode, IQL operates under control of terminal front-end modules, so that terminal users can:

- Write, store, retrieve, or change queries
- Define and interrogate dictionaries
- Define dictionaries reflecting SCHEMA files for DBMS databases
- Browse or update sequential or indexed sequential input files
- Create sequential output files
- Accept raw data input
- Operate other IQL system modules
- Display snapshot reports on the terminal

Deferred Mode

In deferred mode the IQL system provides powerful retrieval selection, report formatting, sorting, computation, summarization, and data file writing capabilities. Up to three input files can be read. The input files can be DBMS database files, sequential files, or indexed sequential files. They can contain fixed- or variable-length records, and one or more record types. Input data files are queried in

Table 9-1 IQL Ad Hoc Reporting

The Query:

```

OPEN CUSTOMER$
RPTHEAD "ANALYSIS OF//PRICE CHANGE//IMPACT"$
IF CSTATE EQ "NH","VT","MA","ME" THEN GO TO
    10 ELSE GO TO NR$
10 COMPUTE XSALES=CYSALES*1.10$
COMPUTE XDIFF=XSALES-CYSALES$
PICTURE XSALES="ZZZZ.99",XDIFF="ZZZZ.00"$
TOTAL CYSALES $ TOTAL XSALES $ TOTAL XDIFF$
PRINT CNAME,1,CYSALES,XSALES,XDIFF,CSTATE$
    
```

The Report:

05-01-81

ANALYSIS OF
PRICE CHANGE
IMPACT

PAGE 1

CUSTOMER NAME	Y-T-D PURCHASES	XSALES	XDIFF
MANFREDINI VIOLIN CO.	\$1,233.67	1357.03	123.36
GLOBALEX	\$2,500.00	2750.00	250.00
LAKESIDE HOMES INC.	\$1,211.00	1332.10	121.10
FARFIELD MOTORS	\$9,000	9900.00	900.00
HUBERT OIL COMPANY	\$600.00	660.00	60.00
BEE DRILL SERVICE	\$1,000.00	1100.00	100.00
INVESTMENT INC.	\$364.75	401.22	36.47
ENERGY RESOURCES INC	\$8,000.00	88000.00	800.00
GEO BANK FISHERIES	\$9,164.00	10080.40	916.40
MERRIMACK VALLEY SALES	\$6,000.00	6600.00	600.00
APPLE ORCHARD INC.	\$671.00	738.10	67.10
GIRAFICS ADVERTISING	\$5,115.50	5627.05	511.55

FINAL SUMMARIES

Y-T-D	TOTAL:	\$44,859.92
XSALES	TOTAL:	49,345.90
XDIFF	TOTAL:	4,485.88

END QUERY EXECUTION

their original format and can be in SIXBIT or ASCII mode. All standard data item types are permitted. Only sequential or indexed sequential input files can be updated; DBMS databases cannot. Sequential output files can be generated that either mirror the format of the primary input data file or assume a new format as specified by the query.

Queries can be stored in text files or in executable form for later reuse.

IQL Statements

An IQL selection statement can combine many conditional tests connected by AND and OR logical operators and clarified with parentheses. An IQL computational statement can include addition, subtraction, multiplication, and division with parentheses. Data item breaks can be used to control built-in summary statements for tally, total, average, maximum, and minimum calculation. A random number variable is built in.

DBMS Files

IQL accesses data files under control of a data dictionary that describes the format of the file and the location, the default display pictures, and the column titles of each item in the data record(s). For DBMS databases the dictionary also includes pertinent information about record names, set names, and area names. Password protection can be applied to individual data items or groups of items.

To use DBMS database files, the SCHEMA file must be present. Privacy locks provide additional security for DBMS databases. IQL's queries operate in stages delimited by SORT statements. IQL can sort data items or calculated fields in ascending or descending order.

Report Formatting

IQL has automatic and custom report formatting. Users specify data item placement in reports. Both multiple print lines per input data record and multiple input data records

per print line are permitted. Special-form reports such as mailing labels and checks can be produced easily. Up to 99 multiple reports can be generated at one time.

SORT/MERGE

The optional TOPS-20 SORT/MERGE utility operates by itself or with COBOL or FORTRAN programs. The user specifies parameters for the sort sequence and the collating sequence.

SORT/MERGE reorders the records of EBCDIC files, ASCII files, SIXBIT files, and binary files produced by COBOL or FORTRAN.

SORT/MERGE automatically controls the use and allocation of disk workspace and memory workspace, although the user can specify memory limits. SORT/MERGE provides error diagnostics and statistics, upon completion.

MERGE combines sorted files into a single sorted file. This function can be used alone, with COBOL, or with FORTRAN.

TRAFFIC-20

Transaction Routing and Forms Filling in COBOL (TRAFFIC-20) is a collection of CRT screen-formatting and program-to-program communications subroutines available to COBOL programs. TRAFFIC-20 is optional on DECSYSTEM-20s.

TRAFFIC-20 includes a set of screen-formatting routines, a set of transaction-routing routines, and a stand alone utility program for defining and saving CRT screen format descriptions. With the screen-formatting routines a COBOL program can send formatted CRT displays to, and receive data messages from terminals. The transaction-routing routines equip a COBOL program to send and to receive data packets to and from another cooperating COBOL program.

COGO-20

COGO-20 is an optional TOPS-20 tool for solving problems in plane coordinate geometry. It includes a geometric language. COGO-20 is used in such tasks as land surveying, highway design, right-of-way surveys, bridge geometry, and subdivision work.

The user works directly with the COGO language, which consists of common engineering terms. No computer experience is required to use it. Using the commands, problem definitions can be stored in a file and executed at a later time, or can be entered interactively on a terminal keyboard.

COGO has commands for:

- Starting and ending a COGO job or changing the I/O device
- Maintaining tables (lists of related points such as road alignments or property lines)
- Intersecting existing lines or figures to calculate a new point
- Calculating and/or storing one or more points
- "Locating" in traverse work

- Alignment and spiral functions
- Maintaining compatibility with the previous COGO so that existing input can be used
- Outputting information generated by COGO

PCS-20

PCS-20 is an optional TOPS-20 Project Control System that analyzes critical path or procedure networks and generates a number of resource, cost, and critical path reports.

PCS-20 can process data for any critical path network developed in an IJ/CPM, or precedence technique. The fixed-format input can be entered on a variety of media. The system includes a terminal-oriented editor for input from a CRT or hardcopy terminal.

PCS-20 has the following features.

- The time required to perform a work item can be expressed in seven different time units.
- Seven different days can be designated as the starting day of the work week associated with each work item.
- The work week of each work item can be one to seven days.
- Any one of three distinct calendars can be used for each work item.
- The actual dates can be specified for each work item.
- One of five types of "scheduled" dates can be specified for the start and completion of each work item.
- Two "early" and two "late" dates can be computed for each work item.
- Two types of duration can be specified for each work item.
- Two types of "percent complete" can be specified for each work item.
- Two kinds of "float" can be computed for each work item.
- Three different kinds of codes can be specified to be associated with each work item.
- Three kinds of costs can be specified for each work item.
- An IJ/CPM or a precedence network can be accepted for processing.
- An IJ/CPM network is internally converted into a precedence network for processing.
- Three different relationships can be specified between any two work items when using precedence input.
- Loops are detected and identified.
- A time delay can be specified that becomes an intrinsic part of the relationship between two work items.
- All network calculations are based primarily on three calculation dates.
- The specification of 240 "milestones" is possible.
- Fourteen different types of output reports can be produced on request:
 - four different kind of system runs can be specified, and
 - a network can extend over a period of 2,911 calendar days, that is, slightly less than eight years.

10 Communication



A range of communications software expands the DECSYSTEM-20's capabilities for interactive computing and distributed processing. Distributed processing software, remote job entry stations, IBM protocol emulators, and a document transmission package for word processing equip TOPS-20 to communicate with other DIGITAL computer systems and with IBM mainframe computers and remote job entry stations.

Distributed processing networks are formed when two or more computers communicate with each other to exchange information. DIGITAL has developed two types of network products — DECnet for communication between DIGITAL computers, and Internets for communication between DIGITAL systems and other vendors' systems.

Using DECnet, computer system networks can be constructed to facilitate remote communications and distributed computation. DECnet is highly modular and flexible. It is a set of tools and services from which a user selects whatever is appropriate to build a network that will satisfy the requirements of a particular application or application set.

DIGITAL Network Architecture (DNA) provides the common network structure upon which all DECnet products are built. The architecture is designed to handle a broad range of application requirements because all the functions of the network — from the user interface to physical link control — are completely modular. DNA allows nodes to operate as switches, front ends, terminal concentrators, or hosts.

DECSYSTEM-20 distributed processing capabilities can be extended with Internets or with PDP-11-based remote job entry stations.

TOPS-20 supports the 2780/3780 and the 2780/3780/HASP protocol emulators. These two Internet products enable DECSYSTEM-20s to communicate with IBM systems. Both perform emulation to communicate with an IBM 360 or 370 host, and termination to communicate with an IBM or IBM-type remote job entry station. These emulators are well suited for batch-mode BISYNC communications. TOPS-20 2780/3780/HASP is a functionally enhanced 2780/3780. Besides cardreader, punch, and printer, it supports a terminal with CRT and keyboard. With it, operators can communicate directly with the IBM mainframe from a local terminal to control and check the status of jobs on the IBM host. DIGITAL Internet products offer distinct advantages over the IBM products whose protocols they emulate.

Users can also install and support a DX/TOPS-20 document transmission package for word processing systems. DX/TOPS-20 links the data processing and file management capability of a DECSYSTEM-20 with the editing and list processing capabilities of DIGITAL's WPS-8 word processing systems.

DECNET-20

DECnet expands the power of DIGITAL computer systems so that each system can be used both independently, and as part of a network. Each can also provide additional computing benefits. There is a DECnet product to support each of DIGITAL's major operating systems.

DECnet-20 equips a DECSYSTEM-20 for use in a computer network that can include other DECSYSTEM-20s, VAX/VMS systems, and PDP-11 systems running the RSX-11, IAS, RSTS/E, CTS-500, or RT-11 operating systems. Users access the network through the standard TOPS-20 file system interface via MACRO-20 system calls. DECnet-20 offers task-to-task communication, network file transfer, and network control capabilities.

Task-to-task communication permits user tasks running on one system to communicate interactively with programs or tasks running on another. The local task uses the TOPS-20 file system monitor calls to operate on files on network devices. Messages sent and received by user programs can be in any data format.

DECSYSTEM-2040s and 2060s can support up to eight physical links over serial synchronous communications lines that can each support several logical links. DECSYSTEM-2020s can support two such links.

DECnet-20 network file transfer utilities permit a user to transfer sequential ASCII files between network systems. Files can be transferred in either direction between the locally supported DECSYSTEM-20 devices and the file system of an adjacent network system.

The DECnet-20 Network Control Program (NETCON utility) supports recording and displaying statistics and load-

ing and dumping the DN20 communications front end or DN200-based RJE station (see *NETCON* below). With NETCON, an operator can display DECnet activity at the local site. The user can display statistics related to communications lines, including data on traffic and errors. Output can be directed to the terminal or to a report file. Other utilities provide controlled local loopback test arrangements that enable users to perform a series of logical tests to aid in isolating communication problems.

When a TOPS-20 system operates as a node in a DECnet configuration, the functions it can perform depend somewhat on other DECnet products in the configuration. This is because two communicating nodes are always restricted to their common capabilities. Table 10-1 lists the various DECnet systems that can be configured with DECnet-20 and the functions each can perform. Functions in the left-hand column are described in the section titled *DECnet Functions* below.

DECnet-20 also includes onsite configuration tools and a set of programs with which the DECSYSTEM-2040 or 2060 user can build site-specific, tailored software according to a DIGITAL-supplied "cookbook" procedure. This software will run in the communications front end of a DECSYSTEM-20 or in a DN200-based RJE station.

DIGITAL Network Architecture

DECnet includes a set of layered network protocols, each of which performs specific network functions. The protocols govern the format, control, and sequencing of message exchange in all DECnet products. Collectively, these protocols are known as the DIGITAL Network Architecture (DNA), and they provide a modular design for DECnet. The

Table 10-1 DECnet Products and Functions

Product (Version)	DECnet-11M (3.0)	DECnet-11M -PLUS (1.0)	DECnet-11S (3.0)	DECnet-IAS (2.1)	DECnet/E (1.1)	DECnet-RT (1.1)	DECnet-VAX (1.3)	DECnet-20 (2.0)
Task-to-Task	yes	yes	yes	yes	yes	yes	yes	yes
Intersystem File Transfer	yes	yes	no	yes	yes	yes	yes	yes
Network Command Terminal	yes	yes	yes*	no	no	no	no	no
Batch/Command File Submission — Requester	no	no	no	yes	yes	yes	no	no
Batch/Command File Submission — Server	yes	yes	no	yes	yes	no	yes	yes
Batch/Command File Execution	yes	yes	no	yes	yes	yes*	yes	yes
Remote File Access	yes	yes	yes*	yes	yes†	yes	yes	no
Downline System Loading	yes	yes	no	yes	no	no	yes	yes
Downline Taskloading	yes	yes	no	yes	no	no	no	no

* requester only

† server only

major protocols are DIGITAL Data Communications Message Protocol (DDCMP), Network Services Protocol (NSP), and Data Access Protocol (DAP).

DIGITAL Data Communications Message Protocol — DDCMP handles the physical link and physical-link error recovery within DECnet. DDCMP operates either full- or half-duplex with serial synchronous or asynchronous facilities in a point-to-point mode. DDCMP:

- Operates over a wide variety of hardware types
- Makes efficient use of full-duplex channel capacity
- Allows transmission of all data types (including binary)
- Allows use of standard character-oriented communications hardware
- Uses CRC-16 for error detection, with recovery by retransmission
- Is effective on satellite links and other links with long signal propagation delays

Network Services Protocol — NSP handles logical link management functions within DECnet. NSP makes it possible for two programs on different machines to establish a logical communications channel (logical link) between the programs and to exchange data using this logical link. These programs need not be aware either of the nature of the physical link (full- or half-duplex, parallel, or serial) or the nature of the protocols supporting the physical link. NSP has:

- Dynamic creation of logical links between tasks
- Exchange of data between tasks on a solicited basis
- Exchange of data between tasks on an unsolicited (e.g., interrupt) basis
- Ability to connect nodes dynamically within the network once NSP initialization occurs over a previously established physical link

Data Access Protocol — DAP enables programs on one node of the network to use the I/O services available on other nodes. Some DECnet products provide facilities for translating operating-system-specific I/O calls into the DAP standard and vice versa. Thus, DAP enables data requests to be processed in a meaningful way by many possibly heterogeneous operating systems. DAP's facilities include remote file access including OPEN, READ, WRITE, CLOSE, and DELETE for sequential and random access files and for command files.

Each DAP function requires support at both ends of the link. At the local node where the user program initiates a data request, DAP must package the request for transmission through the network. At the remote node where the device or file resides, DAP causes the appropriate actions. Not all systems support both local and remote portions of each DAP operation.

DECnet Functions

DIGITAL Network Architecture, implemented across a wide range of operating systems and hardware configurations, enables users to build a variety of networks. While such networks have a common attribute, individual systems within the network can have certain system-specific attributes. As indicated in Table 10-1 the common attribute is *task-to-task communication*. Programs or tasks on one system can create logical links and, in realtime

fashion, exchange data with programs or tasks on other systems.

Table 10-1 also shows that many DECnet systems support other features that are useful in a network environment. These include:

- *Intersystem file transfer* to move an entire data file between systems at either program or operator request. Sequential ASCII files are supported by all the systems and are used in these transfers.
- *Network command terminals* to allow local users to log onto like systems in the network as though their terminals were directly connected to the remote system.
- *Batch/command file submission* to submit batch or command files to remote systems for execution.
- *Batch/command file execution* to allow remote users to submit for execution at the local node a batch or command file that resides at a remote node.
- *Remote file access* for tasks or programs to access sequential files on remote nodes on a record-by-record basis.
- *Downline system loading* to allow initial memory images for DECnet-11 systems in the network to be stored on the local system and loaded on request into other systems in the network. Remote systems usually require the presence of a network bootstrap loader, implemented in read-only memory.
- *Downline task loading* to allow programs executable on DECnet-11S systems in the network to be stored on the local system and loaded on request into other systems under the joint control of the operating systems at both ends of the physical link. This feature and those listed above simplify the operation of network systems that do not have mass storage devices.

Table 10-1 provides the information for determining if the preceding functions are available on a particular DECnet system. The descriptions above are the minimum capabilities provided by a given function. Additional capabilities may be available between two DECnet-connected systems.

DECnet-20 Capabilities

DECnet-20 provides the TOPS-20 user with network task-to-task and file transfer capabilities. Local system or user tasks can exchange information with system or user tasks running in one or more adjacent nodes in a network. TOPS-20 also provides file transfer between systems via the Network File Transfer Program using the COPY, DELETE, DIRECTORY, and SUBMIT commands.

A local task communicates by using the TOPS-20 file system monitor calls to open files, read and write information, and close files on a pseudodevice representing the network.

Below this user layer and transparent to the user, the network protocol takes over. NSP software running in the main processor (and in the communications front end when it is part of the system) manages the actual transfer of data over a logical link. User data is first reformatted into network-compatible segments and then transferred over the logical link. NSP also generates the appropriate control messages to open and close network connections.

NETCON — The NETCON utility (see above) is started at system initialization and runs under SYSJOB. NETCON accepts operator commands which can:

- Downline-load network software into the communications front end of a DECSYSTEM-2040 or 2060
- Load network software into the synchronous line controller of a DECSYSTEM-2020
- Upline-dump the memory of a communications front end of a DECSYSTEM-2040 or 2060 into a TOPS-20 file
- Log pertinent network load, dump, and line status information into the TOPS-20 SYSERR file
- Display line statistics for any active node in the network
- Start, stop, and loop-back lines on a DECSYSTEM-2020

DECnet-20 Operator Interface — Operator commands to NETCON fall into two classes: generic DECnet commands and DECnet-20 commands.

Some DECnet commands are similar on all DECnet computer systems. These are the LOAD NODE and DUMP NODE commands that downline-load and upline-dump the communications front end, the SHOW QUEUE command that checks the status of any queued NETCON requests, and the SHOW COUNTS command that displays linecounter information.

TOPS-20-specific commands are extensions to the generic DECnet commands necessary for implementation under TOPS-20. On the DECSYSTEM-2040 or 2060, the SET NODE command sets up operating characteristics of a network node, and the SET SECONDARY- or TERTIARY-LOAD-FILE command specifies the secondary or tertiary

bootstrap programs to be used for downline loading. On the DECSYSTEM-2020, the LOAD LINE and DUMP LINE commands load and dump the internal line drivers. The INITIATE LOGGING, TERMINATE LOGGING, and SET LOGGING INTERVAL commands control the recording of DECnet-20 line statistics.

DECnet-20 User Interface — With the network file transfer utility (NFT), users can conveniently copy files to or from a remote system, get a directory display for a specified account on a remote system, and submit a job into the remote system's batch queue.

REMOTE JOB ENTRY STATIONS

TOPS-20 supports PDP-11-based remote job entry stations. TOPS-20 support programs running on the DECSYSTEM-20 communicate with the remote PDP-11, using a DECnet-20 task-to-task link. Each station supports up to two high-speed synchronous lines, one lineprinter, and one cardreader. Tasks running in the remote station drive the lineprinter, cardreader and the console terminal.

Remote stations can be downline-loaded with the system software and diagnostics from the host computer. The stations provide the user with remote job entry (remote batch) capabilities to the DECSYSTEM-20. The remote stations have power-fail and automatic restart capabilities.

INTERNET PROTOCOL EMULATORS

The interconnection of DIGITAL systems and computers built by other manufacturers is supported by a family of products called Internets. TOPS-20-supported Internets

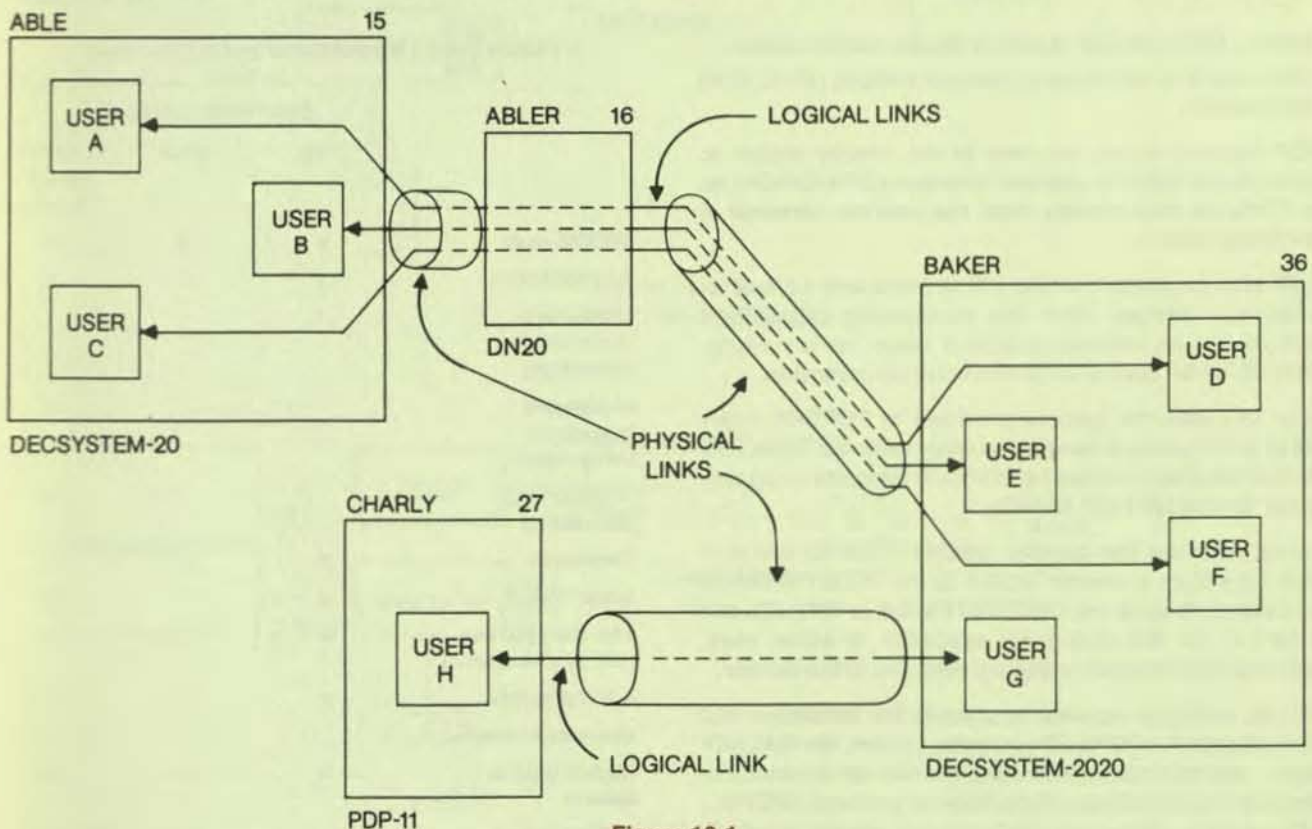


Figure 10-1
Logical and Physical Links

permit reliable bisynchronous links between DECSYSTEM-20s and IBM mainframe systems.

The TOPS-20 Internet products perform both emulation and termination of IBM remote job entry stations. In emulation the software communicates with an IBM 360 or 370 host; in termination the software communicates with a remote station.

The DECSYSTEM-20 Internet products are TOPS-20 2780/3780 and 2780/3780/HASP protocol emulators. The 2780/3780 protocol supports up to six synchronous communications lines on a communications front end connected to a DECSYSTEM-2040 or 2060. Up to two synchronous lines are supported using the HASP protocol.

The TOPS-20 Internets:

- Allow submission from a DATA 100 remote station of card decks containing TOPS-20 batch commands for execution by the TOPS-20 host
- Allow a TOPS-20 user to submit card deck images containing IBM JCL commands for processing by an IBM 360 or 370 remote host running one of the following operating systems:

2780/3780	HASP
OS/VS2 (SVS) HASP II	OS/VS2 (SVS) HASP
OS/VS2 (MVS) JES2	V4
OS/MVT HASP II	OS/VS2 (MVS) JES2,
OS/MVT ASP	NJ
OS/VS2 (SVS) ASP	OS/VS2 ASP
	OS/VS2 JES2

- Allow a TOPS-20 user to print a file at a remote station
- Have features that support remote stations (2780/3780 termination)

HASP support allows the user at the remote station to communicate with the operator interface (OPR/ORION) on the TOPS-20 host directly from the operator terminal at the remote station.

HASP also supports multiple I/O streams and several simultaneous devices. With this multileaving capability a short job can be interleaved while a longer job is running. TOPS-20 HASP supports full character compression.

Table 10-2 lists the features provided by TOPS-20 Internets in termination of remote job entry stations. Table 10-3 lists the features provided by TOPS-20 Internets when emulating remote job entry stations.

Internet software can transfer DECSYSTEM-20 GALAXY batch jobs from a remote station to the DECSYSTEM-20 and execute them at the DECSYSTEM-20, or IBM JCL can be sent to an IBM system for execution. In either case, generated log files and output are returned to the sender.

DIGITAL software required to support this emulation and termination is the TOPS-20 operating system, the GALAXY system, and the Internet software. All Internet communications use the IBM Binary Synchronous protocol (BISYNC, BSC) in half-duplex mode. BSC protocol permits only half-duplex operation.

Table 10-2 Remote Job Entry Termination

	Support Protocol		
	2780	3780	HASP Multi- leaving
EBCDIC code	x	x	x
Extended retry	x	x	x
Multirecord transmission (selectable)	x		
Multirecord interleaved transmission			x
Transparency (selectable)			x
Cardreader	x	x	x
Lineprinter	x	x	x
120-character line	x	x	x
144-character line (132 positions used)	x	x	x
Operator's console			x
Modem control options	x	x	x
Unified system interface	x	x	x

Table 10-3 Remote Job Entry Emulation

	Emulated Protocol		
	2780	3780	HASP Multi- leaving
EBCDIC code	x	x	x
Extended retry	x	x	x
Multirecord transmission (selectable)	x		
Multirecord interleaved transmission			x
Transparency (selectable)			x
Cardreader	x	x	x
Lineprinter	x	x	x
144-character line (132 positions used)	x	x	x
ASCII graphics	x	x	x
Operator's console			x
Modem control options	x	x	x
Unified system interface	x	x	x

In emulation, the Internet software makes the DECSYSTEM-20 appear to be 2780/3780 remote station when it communicates with an IBM host. In termination, it does the reverse — the DECSYSTEM-20 appears to be an IBM 360 or 370 while communicating with a remote batch station.

Consider a DECSYSTEM-20 connected to a remote station by dialup modems. The DECSYSTEM-20 is running under the TOPS-20 operating system and has a DN20 communications front-end processor in addition to the usual console front-end processor. The communications front-end contains the Internet software. In a few steps, the remote station user can send a GALAXY batch job to execute on the DECSYSTEM-20 and receive output returned to the remote station printer.

DX/TOPS-20 DOCUMENT TRANSMISSION

DX/TOPS-20 is an optional, user-supported FORTRAN software package that enables a WPS-8 word processing system to communicate with the TOPS-20 system through an asynchronous terminal interface. The WPS-8 system appears to the host DECSYSTEM-20 to be a normal terminal. DX/TOPS-20 permits distributed stand-alone WPS-8 systems and the host TOPS-20 system to be linked together for better system utilization and data sharing. The

DX/TOPS-20 package includes utility programs that convert TOPS-20 files stored in word processing format to TOPS-20 files stored in ASCII format, and vice versa.

Communications between the WPS-8 system and the TOPS-20 system use the DX error-correcting message protocol.

DX/TOPS-20 lets the WPS-8 system user:

- Store word processing format files on a TOPS-20 system and later retrieve them. By using the DECSYSTEM-20 mass storage, the WPS-8 user can access a large number of documents that might fill several document floppy disks, including large documents filling an entire floppy.
- Use the high-speed DECSYSTEM-20 lineprinter connected to the TOPS-20 system to print rough drafts and other documents that do not require letter-quality printing. This increases system throughput and the availability of the WPS-8 letter-quality printer for printing final drafts. The TOPS-20 system user can output text files converted from the TOPS-20 file system on the WPS-8 letter-quality printer.
- Create TOPS-20 source language and data files offline using the WPS-8 editor, and then transmit the files to the TOPS-20 system for processing.

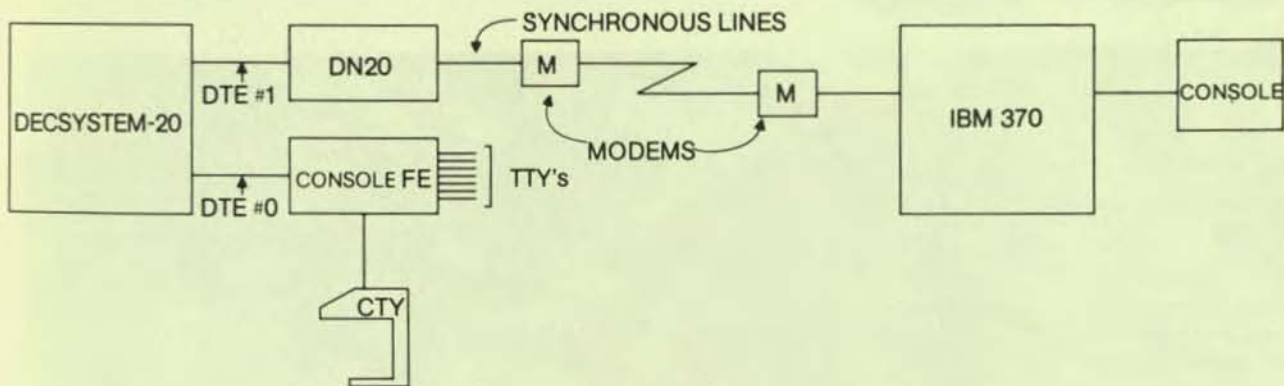


Figure 10-2
2780/3780 Emulation

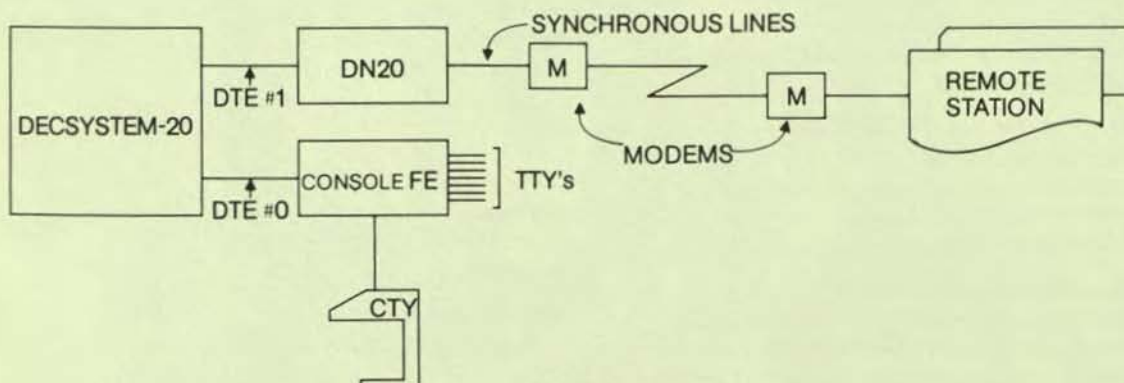


Figure 10-3
2780/3780 Termination

11 Support Services



DIGITAL offers comprehensive support services to help customers before, during, and after system installation. DIGITAL's sales force is the primary contact for all products and services.

The support DIGITAL provides customers is apparent from the beginning. Our sales representatives work closely with customers. They study the application with the customer and determine specific computing needs. Software and hardware specialists are available to supplement the sales representative's product knowledge. These specialists, trained to design systems using DIGITAL's standard and special products, can be called in to answer specific questions.

Once the exact system requirements have been determined, the sales representative helps the customer select a system configuration. Site requirements such as adequate floor space, electrical capacity, air conditioning, and humidity control are reviewed. Customers can choose among various Field Service and Software Service maintenance plans to suit individual needs and budgets.

If the application is complex, the customer and DIGITAL's Software Service organization can prepare a Customer Support Plan (CSP). The CSP can consist of Software Product Services, Educational Services courses, hardware maintenance requirements, and software consulting services. It identifies the customer's needs; the purposes, benefits, and details of the services recommended; how the services will be delivered; and how much they will cost.

Even before the DECSYSTEM-20 arrives, customers can train their personnel through DIGITAL's comprehensive educational programs. When a system is purchased, customers obtain training credits that they can apply to the cost of courses.

When a system is delivered, DIGITAL's hardware Field Service and Software Support organizations are on hand to ensure smooth installation. DIGITAL's field specialists install hardware and software and run tests to determine that the system has been installed correctly and performs properly.

Following installation, DIGITAL's support organizations are available to help with special needs that may arise both during and after the warranty period.

INSTALLATION

Upon delivery of the system DIGITAL's Field Service account representative schedules installation of the hardware components. During installation Field Service engineers supervise the uncrating and placement of equipment, cable connection, and power-up of components. They test the hardware by running a diagnostic package and, once hardware reliability is confirmed, they coordinate with software support personnel to install and test the operating system.

Finally, DIGITAL Field Service and Software Services complete forms that certify successful installation, and the customer acknowledges system acceptance by signing the Field Service Labor Accounting and Reporting System form.

SOFTWARE SERVICES

DIGITAL's Software Services organization specialists are committed to maintaining high-level support for TOPS-20 software. They have the knowledge and experience necessary to analyze the user's needs and to identify and deliver the DIGITAL services that will help satisfy those needs. In addition to local Software Specialists, backup support from regional and corporate levels is available when necessary. DIGITAL's total software resources and expertise are available to support TOPS-20 and its various dependent products.

Software Warranty

TOPS-20 is a DIGITAL-supported software product engineered according to corporate quality standards. It operates in accordance with its Software Product Description (SPD), and carries DIGITAL's commitment to provide support services for the product. TOPS-20 must be installed by a qualified DIGITAL representative to qualify for software support.

DIGITAL-supported software products receive a 90-day warranty following installation. If, during the 90 days of warranty, a problem with the software is encountered that DIGITAL determines to be caused by a defect in the current, unaltered release of the product, the following remedial services are provided.

- If the software is inoperable, DIGITAL will apply a temporary correction or make a reasonable attempt to develop an emergency by-pass.
- DIGITAL will help the customer prepare a Software Performance Report (SPR). Using an SPR, users can report problems with, or suggest enhancements to, DIGITAL's software or documentation.

After the initial purchase of a DIGITAL-supported product license, additional copies can be purchased. These can include support services or can be purchased as a "License-to-Copy only," in which case neither media nor support services are included.

TOPS-20 includes the standard services defined in the TOPS-20 SPD. See your sales representative for these details.

Software Product Services

After the 90-day warranty period, two levels of Software

Product Services are available to provide continued software support and maintenance. Designed to complement DIGITAL hardware services, these Software Product Services offer the most comprehensive post-warranty support in the industry:

- **Software Product Updates.** This service is for customers who install their own software. It includes Software Updates on the user's choice of available media and the most recent documentation. Support services are not provided but are available upon request.
- **Self-Maintenance Service for Software.** This contractual service includes SPR forms, a subscription to software product and documentation updates, and a newsletter that provides up-to-date information on the software product.

Professional Services

Whenever expert software assistance is needed, DIGITAL's software consultants are available. These software specialists are experienced designers and writers of custom software who can tailor DIGITAL software to specific needs. Their expertise can be applied to any phase of an application — from analysis through implementation.

Software specialist services are available on a resident or per-call basis.

- **Resident service** is for users who need full-time, onsite support. Resident consultants are particularly useful in new, complex installations or in critical, long-term projects. Residents are available for a minimum of six months; however, arrangements can be made to extend the length of service to suit individual needs.
- **Per-call service** is for customers with irregular or infrequent consulting needs. Per-call (hourly) services are ordered as needed and generally extend from a few days to a few weeks.

You can learn more about DIGITAL's software services by contacting your local DIGITAL sales office.

EDUCATIONAL SERVICES

DIGITAL provides comprehensive educational programs to train users before, during, and after system installation. Instruction in system management, operations, hardware, and software is given by trained specialists at DIGITAL's worldwide training centers. Special onsite training and custom courses can also be arranged.

Course Options

Courses fall into three general categories:

- **Generic Computer Courses.** These provide a technical foundation for personnel whose computer experience is limited.
- **Software Systems Courses.** These are designed to train users, programmers, and operators to efficiently and knowledgeably use DIGITAL's operating systems, languages, and utilities. Courses are available for both beginning and advanced users. The student is assumed to have general computer and programming knowledge.
- **Hardware Courses.** These are designed for customers who intend to service their own equipment or who want a general understanding of the components in their sys-

tem. Courses in general hardware familiarization, hardware troubleshooting, and hardware maintenance are offered.

The courses can be lecture/lab series regularly scheduled to be taught at DIGITAL Training Centers, onsite courses available by arrangement, or packaged courses.

- Onsite courses. Educational Services can conduct group training courses at any convenient location such as a customer's office or a company's training center. Onsite instruction eliminates travel expenses and allows DIGITAL instructors to emphasize points of particular value to individual applications and operations.
- Packaged courses. Designed for students who wish to learn computer fundamentals and TOPS-20 at a self-paced rate, these packaged courses are portable, self-contained, and modular in format. They are available either as audio/visual courses or self-paced instruction (SPI) workbook courses.

Audiovisual (A/V) courses use a combination of film-strip/tape or videotape and workbooks. Among the A/V courses offered are *Introduction to Data Communications* and *Introduction to Digital Logic*.

Self-paced instruction courses use a workbook with explanatory text, examples, and exercises. Among the SPI courses available are *DECSYSTEM-10/20 DBMS Concepts*, *DECSYSTEM-20 Operator Training*, and *Programming in FORTRAN*.

For users with special needs, Educational Services can create a course tailored to unique applications, needs, and schedules. These can be completely new or modifications of existing courses. Custom courses can be conducted at a DIGITAL Training Center or at a user's training center. Contact your sales representative for full details.

TOPS-20 Courses

DIGITAL Educational Services offers a comprehensive software training program designed to address the needs of all levels of TOPS-20 users. The courses consist of both self-paced and lecture/lab courses that stress practical, job-relevant skills required by TOPS-20 users, applications programmers, and system programmers.

There are 12 TOPS-20 courses. Users' job requirements dictate course content, and students can choose courses from the appropriate sequence. Figure 11-1 is a course flowchart. The option to combine lecture/lab and self-paced instruction courses gives students the convenience of learning at their own speeds and at their job sites, and also the benefits of instructor aid and the practical experience obtained in classroom and laboratory sessions.

The following are brief descriptions of the DECSYSTEM-20 courses shown in Figure 11-1.

TOPS-20 User provides a DECSYSTEM-20 software and hardware overview and teaches students how to use the program development features of the system. Disk file organization, system utilities, command language, and the batch system are covered.

TOPS-20 Administration trains students in the skills needed to administer a TOPS-20 system from startup and shutdown to system installation. Disk file organization, management of resources, operation procedures, and system security are also taught.

TOPS-20 Operator teaches operator duties and the basic system procedures: startup, shutdown, recovery, backup, and file restoration.

TOPS-20 Assembly Language Programming covers the DECSYSTEM-20 instruction set and the MACRO Assem-

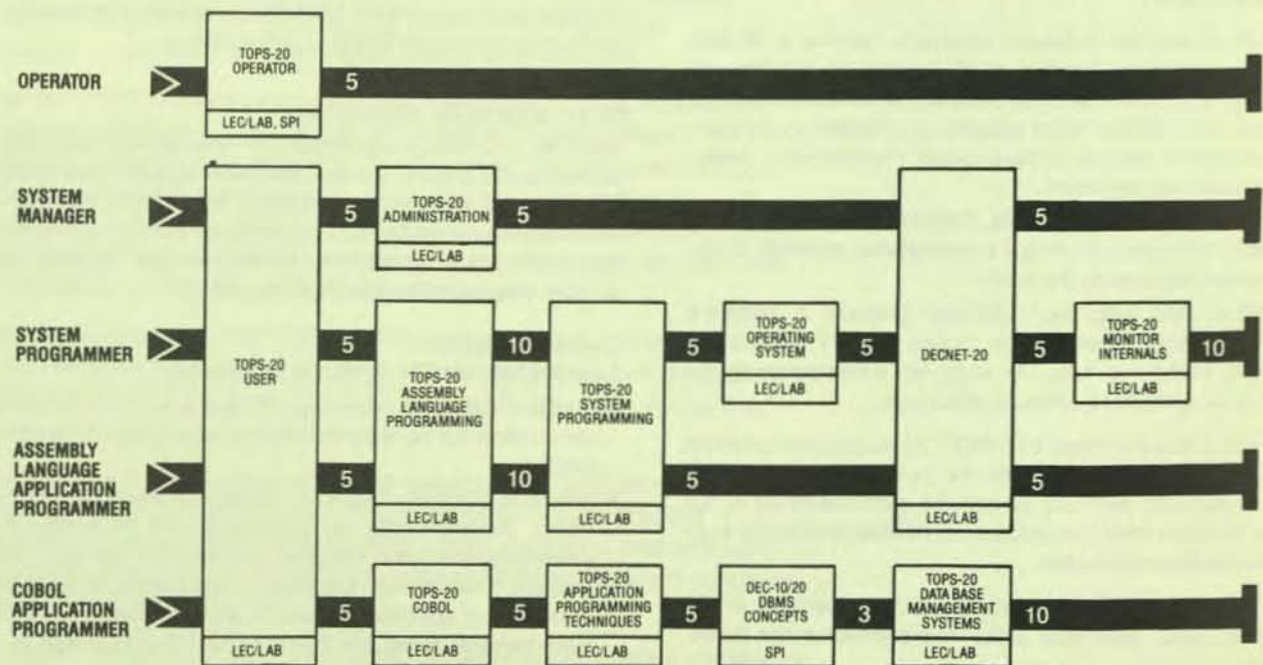


Figure 11-1
TOPS-20 Courses

bler. Students learn to write programs that use monitor calls and the MACRO library and to use Dynamic Debugging Technique.

TOPS-20 COBOL teaches how to write COBOL applications programs under the TOPS-20 operating system. This course assumes the user has prior knowledge of COBOL.

TOPS-20 System Programming covers system controls, the accounting package, and advanced system calls.

TOPS-20 Applications Programming Techniques emphasizes the techniques that can be used to take advantage of system facilities with attention to the best overall design of an application in the timesharing or production environment provided by TOPS-20. This course assumes the user has prior knowledge of COBOL and FORTRAN.

TOPS-20 Operating System teaches the structure and the basic components and functions of the monitor with emphasis on the batch system, command processor, and the monitor's allocation and manipulation of system resources.

DECSYSTEM-10/20 DBMS Concepts is an introduction that covers basic concepts and terminology, compares the inherent TOPS-20 data management system and DBMS data management, and discusses their suitability to various applications.

TOPS-20 Monitor Internals covers the monitor in depth, beyond the structural level, for the programmer responsible for monitor patching and problem detection. Module organization, monitor database, and internal algorithms are stressed.

TOPS-20 Database Management System gives managers, systems programmers, and applications programmers both the conceptual understanding and programming skills necessary to implement DBMS systems.

DECnet-20 presents the network communications products of TOPS-20. Major emphasis is placed on the software required to support network interaction. In particular, the data structures, message formats, and interaction between software modules are described.

HARDWARE SERVICES

DIGITAL's Field Service organization offers a range of post-warranty hardware maintenance services. More than 10,000 Field Service personnel in over 400 locations worldwide are ready to provide the support needed for continuous productivity.

DECservice is DIGITAL's most comprehensive onsite maintenance plan. It is designed for customers who require uninterrupted system performance. DECservice includes:

- Four-hour response time
- Continuous-effort remedial service
- Priority-problem escalation
- Scheduled preventive maintenance
- Parts, labor, and materials
- Installation of the latest engineering change orders
- An assigned service representative

- A comprehensive site management guide

Field Service's other onsite maintenance plan is called Basic Service. Designed for customers who do not require a fixed response time and continuous remedial efforts, Basic Service provides:

- Next-day response
- Remedial service during coverage hours
- Priority-problem escalation
- Preventive maintenance during coverage hours
- Parts, labor, and materials
- Installation of the latest engineering change orders
- An assigned service representative
- A comprehensive site management guide

Although standard coverage for both onsite service plans is eight hours a day, five days a week, customers can opt to extend their service coverage to 12, 16, or 24 hours, including weekends and holidays.

Hardware maintenance on a per-call, time-and-materials basis is also available.

More information is available on DIGITAL's Field Service hardware maintenance offerings from your local DIGITAL sales representative.

COMPUTER SPECIAL SYSTEMS

Computer Special Systems (CSS) provides design services and resulting repeat products to DIGITAL's customers. Analysts, engineers, programmers, and manufacturing specialists can provide hardware and software to customers whose needs are not met by DIGITAL's standard product offerings. In addition, CSS is represented by Marketing Representatives in the field.

CSS provides custom systems and products in all of DIGITAL's markets. Products and systems are analyzed, designed, and implemented according to each customer's goals and requirements. These products can range from simple processor interfaces to complete hardware and software systems. All products carry DIGITAL's high standard of quality, documentation, and field support. CSS design and project management services, available on a contract basis, include:

- Interfacing DIGITAL hardware with that of other manufacturers, modifying standard hardware, and designing and building new equipment
- Designing and producing diagnostic systems and applications software, modifying and expanding standard DIGITAL software systems, or building new software according to individual needs
- Building complete hardware and software systems for special applications. CSS project managers oversee the analysis, design, and implementation of the system and work with the customer to ensure proper installation and start-up

Additionally, customers can receive DIGITAL's broad range of support services with any CSS system. Hardware is supported by DIGITAL's extensive Customer Service organization. Training is available for all aspects of CSS systems.

The CSS solution puts DIGITAL's technology and experience at each customer's fingertips. It provides a cost-effective system designed for a specific application and backs the final product with high-quality service. Contact your nearest DIGITAL sales office for further details.

CUSTOMER FINANCING

To simplify the financial considerations involved in acquiring a new computer, DIGITAL provides leasing and financial counseling. The Customer Finance Department can help customers acquire a DIGITAL system through a lease, conditional sale, or similar financing agreement, rather than an outright cash purchase.

For commercial businesses or private organizations DIGITAL has developed a program known as DIGITAL Leasing with the U.S. Leasing Corporation of San Francisco. DIGITAL Leasing, a division of U.S. Leasing, is committed solely to financing DIGITAL computers. Representatives are located in or near many of the DIGITAL District Sales Offices.

Federal, state, and local government agencies have special contractual needs and, in some cases, can benefit from special tax privileges. For example, a state or municipal agency qualifies for special interest rates on Conditional Sales Agreements, rates that are significantly lower than those charged to commercial customers. The interest income is free from federal and, in some cases, state income taxes.

The following financing is available: Full Payout Lease, Conditional Sales Agreement, and Federal Government Lease to Ownership Agreement.

- Full Payout Lease — This is used primarily by commercial customers. It is noncancellable, lasts three to five years, and usually has a 10 percent purchase option at the end. No down payment is required, and title remains with the lessor. Flexible lease payment schedules can be tailored to specific needs.
- Conditional Sales Agreement — This type of financing is used primarily by state and local governments. It is noncancellable, and lasts one to five years. Title passes to the customer, but DIGITAL retains a security interest. The customer owns the equipment free and clear at the end of the term. Fiscal funding provisions are available for state and local governments.
- Federal Government Lease to Ownership Agreement — This is available only to approved federal government agencies. It lasts one to five years, and is cancellable at the end of each fiscal year for nonappropriation of funds. Ownership passes to the customer at the end of the term.

DIGITAL's Customer Financing group can provide financial counseling to help you decide which arrangement is best for you. For more information, contact your local DIGITAL sales office.

ACCESSORIES AND SUPPLIES GROUP

DIGITAL's Accessories and Supplies Group (A&SG) maintains Accessories and Supplies Centers (ASCs) and offers

direct factory ordering, the *Direct Sales Catalog*, and worldwide support for their products.

ASCs are full-service centers established to fulfill the needs of DIGITAL customers in major metropolitan areas. The ASCs hold a local inventory of the most requested accessories, supplies, documentation, and add-on products, for fast delivery. Full order processing capability provides access to A&SG's central inventory in Nashua, New Hampshire. ASCs provide first-class service and convenience to DIGITAL's customers.

A&SG maintains a tollfree telephone number for customers to use when ordering accessories and supplies. The majority of products are shipped within 48 hours of receipt of order.

A&SG's *Direct Sales Catalog* offers a broad range of computer accessory and supply items. These include small computer systems and their complementary options, accessories, and operating supplies. The *Direct Sales Catalog* also features some DIGITAL hardware options such as DECwriters, microcomputers, and associated options. Hardware and software documentation is also offered.

A&SG has a team of worldwide specialists and business managers to support sales. This sales force is located in the United States, Europe, and the General International Area (GIA).

COMPUTER SUPPLIES

DIGITAL's Computer Supplies group maintains a complete line of supplies specifically designed for use with DIGITAL systems. These items facilitate reliable and efficient system operation and include:

- A family of magnetic media such as disk cartridges, disk packs, and floppy diskettes
- Self-contained disk cartridge cleaners for fast and efficient cleaning of front-loading or top-loading magnetic disk cartridges
- Word processing supplies such as nylon and mylar ribbons, 11 types of print wheel, and filter screens for video terminals
- Ribbons for DIGITAL's DECwriter and DECprinter terminals

The Computer Supplies group also offers computer cabinetry for maintaining supplies and protecting magnetic media not in use. Cabinet interiors can be customized with various options to meet individual needs. Options can be conveniently rearranged or changed at any time to adapt to future requirements. Also available are paper baskets, work shelves, terminal tables, tape racks, paper-tape trays, and multipurpose binders.

Relying on DIGITAL for computing needs means a savings in time, money, and paperwork. Contact your sales representative for further information.

CUSTOMER SPARES

Customer Spares is dedicated to supporting customers who maintain their own computers. Customer Spares is organized into three distinct businesses: self-maintenance

products (hardware and documentation), system accessories, and low-volume LSI-11 products. System accessories include products geared to the hardware builder. They allow easy expansion and reconfiguration of DIGITAL systems and options.

Customer Spares sells modules, subassemblies, components, tools, and test equipment. Related services involve providing assistance in selecting the proper parts and expediting delivery during emergency situations.

DECUS

DECUS, the Digital Equipment Computer Users Society, is one of the largest and most active user groups in the computer industry. It is a not-for-profit association supported and administered by DIGITAL but actively controlled by individuals who have purchased, leased, ordered, or used a DIGITAL computer or who have a bona fide interest in DECUS. Membership is free and voluntary.

The goals of DECUS are to:

- Advance the art of computation through mutual education and the exchange of ideas and information
- Establish standards and provide channels that ease the exchange of computer programs
- Provide feedback to DIGITAL on hardware and software needs
- Advance the effective use of DIGITAL computers, peripherals, and software by promoting the interchange of information



DECUS headquarters, located in Marlboro, Massachusetts, administers all international policies and activities. DECUS is subdivided into four chapters:

**DECUS AUSTRALIAN
CHAPTER** (Australia, Brunei,
New Zealand, Malaysia,
Singapore, Indonesia, PNG)
DECUS Australia
P.O. Box 384
Chatswood
NSW 2067
Australia

**DECUS CANADIAN
CHAPTER**
DECUS Canada
P.O. Box 13000
Kanata, Ontario
K2K, 2A6, Canada

**DECUS EUROPEAN
HEADQUARTERS** (Europe,
Middle East, North Africa,
Eastern Europe)
DECUS Europe
P.O. Box 510
12, Av. Des Morgines
CH-1213 Petit-Lancy 1/GE
Switzerland

**DECUS UNITED STATES
CHAPTER** (for U.S. and all
others)
DECUS International
Headquarters
Digital Equipment
Corporation
MR2-3/E55
One Iron Way
Marlboro, MA 01752 U.S.A

To further the goals of the society, DECUS serves its members by holding symposia; maintaining a program library; publishing an association newsletter, technical newsletters, and books; and supporting a number of special user groups for special interests and locations.

- **Symposia** — These are regularly scheduled meetings held in each of the four chapters. They provide a forum in which users of DIGITAL products can meet with other users and with DIGITAL management, engineering, and support personnel. Symposia give users an opportunity to participate in DIGITAL product workshops and product planning feedback sessions. Many of the technical papers and presentations from each symposium are published as a book, the *DECUS Proceedings*. Copies of the *Proceedings* are supplied to all symposia attendees and can be purchased by any DECUS member.
- **Program Library** — A major activity of DECUS is the Program Library. It contains over 1,700 active software packages written and submitted by users. A wide range of software is offered including languages, editors, numerical functions, utilities, display routines, games, and other types of application software. Library catalogs are available for the PDP-8, PDP-11/VAX, DECSYSTEM-10/20. Catalogs are updated yearly and contain program descriptions and ordering information. The programs are available at nominal service charges that cover the cost of reproduction and media.
- **Association Newsletter** — Each DECUS chapter publishes and distributes to all chapter members a newsletter of general DECUS news.
- **Special Interest Groups** — The focus of these groups is on operating systems, languages, processors, and applications. User groups are formed basically by geographical proximity, but can also share common specific interests. Many of these subgroups publish newsletters.

You can obtain a membership form for DECUS by contacting the appropriate chapter office.

The Glossary

glos·sa·ry (gl
basic technical,
ject or field, wit
glossa GLOSS²]
—glos/sa·rist,

Absolute address A binary number assigned as the address of a fixed memory storage location.

Absolute virtual address A fixed location in user virtual address space which cannot be relocated by the software. However, it can still be translated to a physical address by the hardware.

AC Refer to *Accumulator*.

Access privileges Attributes of a file which specify the class of users allowed to access the file and the type of access they are allowed.

Access time The time between requesting a data transfer with a storage device and transferring the first bit.

Account name An identifier that indicates a user account for both file structures and file access.

Account validation A program within the operating system that is used to verify account names.

Accumulator A hardware register and its associated equipment in the arithmetic unit in which data can be placed while the data is being examined or manipulated.

Active process A process considered by the operating system to be runnable.

Address 1. An identification represented by a name, label, or number for a register, a location in storage, or any other data source or destination in memory or on an addressable storage device. 2. The part of an instruction that specifies the location of an operand of the instruction.

Address constant A value or expression used in calculating absolute or virtual storage addresses.

Addressing Means by which locations in storage are specified or computed. Addresses may be specified as absolute numbers or may be computed using absolute values combined with the contents of registers. The address computed may be used in an instruction or it may indicate the location of the address to be actually used in the instruction.

Address mapping The assignment of user virtual address space to the physical address space in computer memory. This is automatically performed by the monitor and is completely invisible to user programs.

ALGOL ALGOritmic Language. A scientifically oriented language that contains a complete syntax for describing computational algorithms.

ALGOTS The ALGOL Object Time System.

Alphanumeric Pertaining to the characters which include all the letters of the alphabet and the numerals 0 through 9.

ALTMODE A terminal key which is used to end some command strings. See also *ESCAPE*.

Analog Dealing in continuous variations or gradations, as contrasted to discrete (in steps) variations. Pertaining to data in the form of continuously variable physical quantities.

AND A logical operation such that the result is true if all conditions are true; otherwise the result is false.

ANSI American National Standards Institute. An organization that develops and publishes industry standards.

APL A Programming Language. A problem-solving language with special features for handling arrays and for performing mathematical functions.

Application program A program written for or by users that applies to their own work.

Archival Storing a file of unchanging data on magnetic tape.

Argument An independent variable. For example: 1. A variable or constant which is given in the call of a subroutine as information to it. 2. A variable upon whose value the value of a function depends. 3. The known reference factor necessary to find an item in a table or array (i.e., the index). Contrast with *subcommand*.

Arithmetic unit The component of a computer where arithmetic and logical operations are performed. Part of the central processor.

ARPANET Advanced Research Project Agency Network

Array An arrangement of elements in one or more dimensions (i.e., an ordered arrangement of subscripted variables).

ASCII American Standard Code for Information Interchange. A 7-bit code in which textual information is recorded. ASCII can represent 128 distinct characters. These characters are the uppercase and lowercase letters, numbers, common punctuation marks, and special control characters.

ASCII character set The set of 128 7-bit ASCII characters.

Assemble To translate from a symbolic language program to a machine language program by substituting binary operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.

Assembler A program which translates symbolic programs into machine language programs and assigns memory locations for variables and constants.

Assembler directive The mnemonic used in the assembly language program to control or direct the assembly process.

Assembly language A symbolic language that translates directly into machine language instructions. Usually there is one-to-one relation between assembly language instructions and machine language instructions.

Assembly listing A printed list which is an optional by-product of an assembly run. It lists in sequence the symbolic code written by the programmer next to a representation of the actual machine instructions generated. The contents of the assembly listing are mostly under the control of the programmer.

Assembly time The time at which the assembler translates the symbolic machine language statements into machine instructions.

Assigning a device Associating an I/O device with the user's job either for the duration of the job or until the user relinquishes it.

Asynchronous 1. Pertaining to the procedure by which the hardware can begin a second operation before waiting for the first operation to be completed. 2. Pertaining to the method of data transmission in which each character is sent with its own synchronizing information and no fixed time between consecutive characters.

Background program A program operating automatically at low priority, when higher priority (foreground) programs are not using system resources.

Backspace To move backwards the logical position in a file or on a line according to a prescribed format. For example, magnetic tape units can be backspaced over a file or a record. Some terminals allow backspacing to permit overprinting.

Backup file A copy of a file created for protection in case the primary file is inadvertently destroyed.

Balance set The set of processes eligible to run and whose working sets fit into memory simultaneously; periodically redefined by the scheduler.

Base address An address used as a basis for computing the value of some other address. This computation is usually of the form: final address = base address (+ or -) offset.

BASIC Beginner's All-purpose Symbolic Instruction Code. An easily learned interactive computer programming language.

Batch mode A system state in which programs are run consecutively without terminal interaction, as opposed to timesharing.

Batch processing The technique of executing a set of computer programs without human interaction or direction during their execution.

Batch stream A collection of batch command strings which are executed successively without the intervention of an operator.

BATCON The Batch Controller. This program reads a job's control file, starts the job, and controls the job by passing commands and data to it.

Baud A unit of signaling speed equal to the number of

signal events per second. Baud is the same as bits per second when each signal event represents one bit.

Baud rate A fixed amount of time devoted to sending a binary digit or bit.

Binary A characteristic, property, or condition in which there are but two possible choices; for example, the binary number system, using 2 as its base, has the digits 0 and 1.

Binary code A code that uses two distinct characters only, usually 0 and 1.

Bit map A table describing the allocation of space or resources. For example, each bit in the table indicates the state of one segment of storage, as a block on a bulk storage device.

BLISS-36 A programming language that enables users to write programs consisting only of declarations that establish structure, and expressions that compute values. It is designed for implementing system software.

Block (in a program) A segment of a program or a group of instructions in a program that performs some function that is a logically distinct subtask of the entire job.

Bootstrap A routine or device designed to bring itself into a desired state by means of its own action, e.g., a machine routine whose first instructions are sufficient to bring the rest of the routine into the computer from an input device.

Bootstrap loader A routine whose first few instructions are sufficient to bring the rest of the routine into the computer from an input device.

Branch A transfer of program control instruction execution from one place in storage to another.

Breakpoint A location where program operation is suspended to examine partial results. Breakpoints are used in the debugging process.

Buffer A temporary storage area that may be a special register or an area of storage. Buffers are often used to hold data being passed between processes or between devices which operate at different speeds or different times.

Bus A circuit over which data is transmitted.

Byte A sequence of adjacent binary bits acted upon as a unit.

Byte manipulation The ability to manipulate, as individual instructions, groups of bits as characters.

Cache memory Very fast semiconductor memory, usually small in size.

Call A branch, usually from one program to another. Usually implied is the ability, via appropriate instructions, to resume the first program from the location just following the point from which the original branch was made.

Card column One of the 80 vertical lines of 12 punching positions on a punched card.

Card field A fixed number of consecutive card columns assigned to a unit of information.

Card hopper The tray on a card processing machine that holds the cards to be processed and makes them available to the card feed mechanism.

CDRIVE The cardreader driver.

Central Processing Unit (CPU) 1. The part of the computer that moves values, does arithmetic, and performs comparisons and other manipulations, as well as keeping track of the current instruction, storage location, and operand addresses. 2. The unit of a computing system that includes the circuits controlling the interpretation and execution of instructions.

Central site The location of the central computer. Used in conjunction with remote communication to mean the location of the central processor as distinguished from the location of the remote station.

Channel 1. A path along which signals can be sent; for example, data channel, output channel. 2. The portion of a storage medium that is accessible to a given reading station; for example, track, band.

Character-oriented Operations performed on a single character.

Checkpoint A place in a routine where a check or a recording of data for restart purposes is performed.

Closed subroutine A subroutine that can be stored at one place and called from one or more calling routines. Contrast with *open subroutine*.

COBDDT The COBOL Dynamic Debugging Technique program.

COBOL Common Business Oriented Language. A programming language used in business data processing applications.

COBOL compiler A program that translates COBOL language statements into machine language programs.

CODASYL Conference on Data SYstems Language. The conference that developed COBOL.

Command The portion of an instruction word that specifies the operation to be performed.

Command language The set of commands with which the user informs the operating system of the desired function.

Command processor A component of the operating system that reads and interprets commands issued by the user.

Command string A group of commands in a single line.

COMMON area A section in a program's address space

which is set aside for common use by many modules. COMMON is usually set up by modules written in the FORTRAN language. It is used by independently compiled modules to share the same data locations.

Communication hardware Devices such as terminals, modems, and acoustic couplers used for transmission, switching, and termination of communication.

Communication link The physical means of connecting one device to another for transmitting and receiving data.

Communication processor A computer dedicated to performing complete communication functions such as message switching and data concentration.

Communication software Software written to explicitly control communication between systems.

Compile 1. To produce a binary-coded program from a program written in source (symbolic) language, by selecting appropriate subroutines from a subroutine library, as directed by the instructions or other symbols of the source program. The linkage is supplied for combining the subroutines into a workable program, and the subroutines and linkage are translated into binary code. 2. To prepare a machine language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one machine instruction for each symbolic statement, or both, as well as performing the function of an assembler.

Compiler A program which translates a high-level source language into a language suitable for a particular machine.

Complement One's complement — to replace every 0 bit with a 1 bit and vice versa. Two's complement — to form the one's complement and add 1.

Computer network A complex of two or more interconnected computers.

Concealed submode A user submode that contains proprietary information.

Concurrent processing The processing of two or more activities within the same interval of time.

Configuration A particular selection of computer, peripherals, and interfacing equipment that function together. A list of the devices and computers of a computer system.

Connect time A measure of system usage by a user, usually the time interval during which the user terminal was online during a session. The wall-clock time between login and logout. See also: *CPU time*.

Console (CTY) The part of a computer used by the operator to determine the status of, and to control the operation of the computer. The console has controls and indicators used for manual operation of the computer.

Context switching The saving of sufficient hardware

and software information of a process so that it may be continued at a later time, and the restoring of similar information relevant to another process. A common example of context switching is the temporary suspension of a user program so that the monitor can execute a monitor call.

Control file The user-made file that directs the batch system in the processing of the user's job.

Control character A character whose purpose is to control an action, such as a line spacing, paging, or termination of a job.

Controller A device or portion of a device that controls the operation of connected units.

CPL Conversational Programming Language.

CPU time A measure of system usage, based on the total amount of central processor time used. See also: *connect time*.

CREF A program which produces a sequence-numbered assembly listing followed by tables showing cross-references for all operand-type symbols, all user-defined operators, or all operation and pseudo-op codes.

Cross-reference listing A printed listing that identifies all references in an assembled program to each specific label, macro, and/or operation code.

CTY Refer to Console.

Cylinder The tracks of a disk storage device that can be accessed without repositioning the access mechanism.

Database A collection of related data items processable by one or more applications.

Database management A systematic approach to storing, updating, and retrieving information stored as data items.

Data channel The device that passes data between the memory system and a device controller.

Data Manipulation Language (DML) The language used by the programmer to cause data to be transferred between a program and the data base. An incomplete language by itself, it requires a host language.

DDT The Dynamic Debugging Technique program used for online checkout, testing, examination, and modification of object programs.

DDCMP Digital Data Communications Protocol. A uniform discipline for transmitting data between nodes in a point-to-point or multipoint communication system. It governs parallel, serial-synchronous, or serial-asynchronous transmission and receipt of data.

Debugging The process of locating any errors in a program, and correcting them.

Default value The choice among exclusive options made by the system when no explicit choice is specified by the user.

Delimiter A character that separates, terminates, and organizes elements of a statement or program.

Demand paging Bringing into memory only those pages that are referenced, and bringing each page into memory when it is first referenced.

Device directory listing A list of all files on a specified device. The list contains all files with associated creation dates, total free blocks on the device, and number of blocks used by files.

Device handler or routine Routines that perform I/O for specific hardware devices. They usually translate logical block numbers to physical addresses for those devices that associate addresses with data. These routines also handle error recovery and provide device independence in operating systems.

Device independence The ability to request I/O operations without regard to the characteristics of specific types of I/O devices.

Diagnostic program A program that facilitates maintenance by detection and isolation of malfunctions.

Direct address An address that specifies the location of an operand. Contrast with *indirect address*.

Directory device A storage retrieval device, such as a disk, that contains information describing the names of files and the layout of stored data. A directory of these files is maintained on the device to locate the files. A directory device must be randomly accessible.

Distributed network A network configuration in which all node pairs are connected either directly, or indirectly through intermediate nodes.

Double precision The use of two computer words to represent a single number.

Dump 1. A listing of variables and their values, or a listing of the values of locations in memory. 2. To copy the contents of all or part of a storage medium, usually onto another storage medium.

Effective address An address computed by taking all the components indicated in a given instruction (such as absolute value, contents of base register or instruction location register, index registers, etc.) and combining them to obtain a single address value.

ENQ/DEQ facility Allows cooperating processes to share resources and facilitates dynamic resource allocation using the ENQ and DEQ monitor calls.

ESCAPE A control key on a terminal that is used for command recognition and to end some command strings.

Exclusive OR A logical operation such that the result is

true if one and only one operand is true, and false otherwise.

EXEC Informal name for the TOPS-20 Command Processor. Contains routines which handle TOPS-20 commands. These commands interact with the monitor.

External interrupt An interrupt caused by a device controller.

Fatal error An error from which a program cannot recover.

FDB File Descriptor Block. Each file has an associated FDB that contains data concerning the file. An FDB may be up to 36 words.

Field A fixed number of consecutive characters assigned to a unit of information.

FILDDT File Dynamic Debugging Technique. A version of DDT used for examining and changing a file on disk instead of in memory. This program is used to examine a monitor for debugging purposes.

File A collection of records. An organized collection of information directed toward some purpose.

File-structured device A device on which data is given names and arranged into files; the device also contains directories of these names. Usually synonymous with directory device.

File descriptors Identifiers that describe an attribute of a file. There are three types of file descriptors: protection numbers, account descriptors, and temporary file descriptors.

File pointer Indicates the last byte read from or written to an open file.

File protection code A code assigned to each file to specify the access rights to the file.

Flag A variable or register used to record the status of a program or device. In the latter case, also called a device flag.

Floating point A method for storing numeric data where one part represents the position of the radix point (the exponent part), and another part represents the significant digits (the fractional part).

Floating-point binary A method for storing numeric data where the exponent occupies the first byte and the mantissa resides in the next three or seven bytes. Used in FORTRAN IV representation.

FORDDT FORTRAN's dynamic debugging tool.

FORTRAN Formula Translator. A procedure-oriented programming language designed for solving scientific-type problems by expressing the procedure for their solution as arithmetic formula.

FOROTS FORTRAN's object time system

Front-end processor The RSX-20F monitor. Controls slow I/O, main memory, and communication devices; transfers data from such to rear-end memory.

Frozen access A type of access in which a file can be concurrently accessed by at most one user writing the file, but by any number of users reading the file.

Frozen process A process whose execution is suspended so that it can be continued where it was suspended.

Full-duplex operation Simultaneous, two-way communication between two points.

General register A register used for operations such as binary addition, subtraction, division, and multiplication. General registers are usually used to modify an address.

Generation number A number associated with every file that indicates the number of revisions of that file that have occurred. Every time the user edits the file, the generation number is typically increased by 1.

Global A value defined in one program module and used in others. Globals are often called entry points in the module where they are defined and externals in the other modules that use them.

Global symbol A symbol accessible to modules other than where it is defined. The value of a global symbol is placed in the loader's global symbol table when the module containing the symbol definition is loaded.

Guide word A word output when you use recognition that indicates the type of argument that you can give for the next field.

Half-duplex Refers to an alternate, one way at a time, independent transmission between two points.

Hardcopy A printed copy of machine output in a visually readable form.

Hardware Physical equipment, as opposed to a computer program or method of use.

Host language A language into which the Data Manipulation Language is integrated to perform actions on the database. The host language, for example, FORTRAN or COBOL, is used to manipulate the data once it is in main memory.

Indexed address An address formed by adding the contents of an index register to the contents of an address field prior to or during the execution of a computer instruction.

Indexing A programming technique in which different locations are addressed by systematically changing the contents of an index register. The program uses the contents of the index register to compute the effective address.

Index register A register whose contents are used in computing addresses. The common function of such a

register is to allow for convenient modification of addresses during program execution.

Indirect address An address which indicates a storage location where the address of the referenced operand or another indirect address is to be found. Contrast with *direct address*.

Inferior process A process created by a superior process with the PUSH command. Inferior processes run under the same job as the superior process that created them.

INFO The central information utility for IPCF. It performs functions associated with names and PIDs.

Input buffer A temporary storage area for input to the computer.

Instruction A bit pattern interpreted by the computer as a direction to take a specific action.

Instruction format The allocation to a specific function of bits or characters of a machine instruction.

Instruction set A set of instructions that a particular computer can perform.

Interleaving To arrange parts of a sequence to alternate with parts of another sequence. The sequence can be things or events, and more than one sequence can be interleaved. In interleaving, each sequence retains its identity.

Internal storage Addressable high-speed storage directly controlled by the central processing unit.

Interpreter A computer program or routine that translates and executes each source language statement before translating and executing the next one.

Interrupt A break in the normal flow of control during the execution of a program. When an interrupt occurs, control can be passed to a specific routine responsible for processing the interrupt. The term interrupt is sometimes used to refer to the condition causing the break instead of the break itself.

Interrupt service routine Routine entered when an external interrupt occurs.

Interrupt vector address A unique address which points to two consecutive memory locations containing the start address of the interrupt service routine and priority at which the interrupt is to be serviced.

IPCF Inter Process Communication Facility. Allows processes to communicate with each other via packets of information or messages.

JFN Job File Number. The unique identifier of a particular file on a particular device during the execution of a process. Used by the system as an index into the table of files associated with the job. Also known as File Handle.

Job A set of one or more related processes normally under control of a single user.

Job number A number assigned to a job at login time.

Job status word A word containing bit flags indicating the status of the program currently in memory.

JSB Job Storage Block. A table which contains all data relevant to a given job.

Kernel mode The executive submode in which I/O and system-wide functions operate. Code executed in kernel mode can access and alter all of memory.

LIBOL The COBOL object time system.

Library A collection of standard routines available to several programs.

Link To combine independently translated modules into one module in which all relocation of addresses has been performed relative to that module and all external references to symbols have been resolved based on the definition of internal symbols.

Linker A program that combines many input modules into a single module prior to loading. Thus, it allows for independent compilations of modules. Typically, it satisfies global references and may combine control sections.

Linking loader A program that provides automatic loading, relocation, and linking of compiler and assembler-generated object modules.

Load module The output of the linker. A program in formatted binary form ready for loading and executing on a computer.

Local terminal A terminal connected directly to the computer. Contrast with *remote terminal*.

Log file A file into which BATCON writes all the information passing between the job and its controlling terminal. This file can be printed as the final step in BATCON's processing of a job.

Logical name An alphanumeric name chosen to represent a physical device and, optionally, a file name. This name can be used in all references to the device.

LPTSPL The lineprinter spooler. Takes files from disk and prints them on the lineprinter.

Macro An instruction in a source language that is equivalent to a predefined sequence of machine instructions, or a command in a command language that is equivalent to a specified sequence of commands.

MACRO-20 A programming language that is closer to machine language than a high-level language such as FORTRAN. Also called an assembly language or assembler.

MAILER A program that allows users to communicate

with each other by sending and receiving messages in the form of data files.

Main program The module containing the address at which object program execution normally begins. Usually, the main program exercises primary control over the operations performed and calls subroutines to perform specific functions.

Mask A bit pattern that selects from a word in memory those bits that are to be used in a subsequent operation. A mask can be used to retain or eliminate portions of any word in memory.

Mass storage An auxiliary storage of very large capacity readily accessible to the CPU.

Memory protection A scheme for preventing read and/or write access to certain areas of storage.

Mode A method of operation.

Monitor 1. The collection of programs that schedules and controls the operation of user and system programs, performs overlapped I/O, provides context switching, and allocates resources so that the computer's time is efficiently used. 2. The master control program that observes, supervises, controls, or verifies the operations of a system.

MOS memory Metal-oxide semiconductor memory.

Multiprocessing The simultaneous execution of two or more computer programs (i.e., processes) by two or more interconnected processors sharing a common, jointly addressable memory.

Multiprogramming The apparent simultaneous execution of two or more programs on a single computer by timesharing system components.

Multitasking The activation of several separate but related tasks under a single program identity.

Named file A name-ordered collection of 36 bits whose length is limited only by the available space on the device and the user's maximum space allotment on that device.

Network control program (NCP) The module of an operating system in a host computer that establishes and breaks logical connections and communicates with the network on one side, and with user processes within the host computer on the other side.

Node Any computer, communication controller, terminal concentrator, or remote station in a computer network.

Nonresident software Programs that reside on the disk until called for by a user or other program.

Nonsharable program A program for which each user has a separate copy.

No op An instruction that specifically instructs the computer to do nothing, except to proceed to the next instruction in sequence.

Null character A control character that accomplishes media fill or time fill.

Object code Output from a compiler or assembler which is itself executable machine code.

Object module The primary output of an assembler or compiler. It can be linked with other object modules and loaded into a runnable program. This output is composed of the relocatable machine language code for the translated module (i.e., link items), relocation information, and the corresponding symbol table listing the definition and use of symbols within the module.

Object time system The collection of modules that is called by the compiled code for a particular language in order to perform various utility functions. This collection usually includes I/O and trap-handling routines. For example, FOROTS is the object time system for the FORTRAN language.

Offline file A file that has been moved from disk to tape by either the virtual disk system or the archive system.

Offline storage Storage not under the control of the CPU.

Offset The number of locations or bytes relative to the base of an array, string, or block.

Operating system Software that controls the execution of computer programs and may provide scheduling, debugging, input and output control, accounting, storage assignment, data management, and related services. Sometimes called the Supervisor, Executive, Monitor, Master Control Program depending on the manufacturer.

OR 1. (Inclusive) A logical operation such that the result is true if one or more operands are true, and false if all operands are false. 2. (EXclusive) A logical operation such that the result is true if one and only one operand is true, and false if otherwise.

Overlay 1. The technique of repeatedly using the same sections of virtual address space to store different blocks of code and data during different stages of a program. When one routine is no longer needed in storage, another routine can replace all or part of it. 2. One of several blocks of code and data that potentially can occupy the same section of virtual address space. 3. A segment of a program that is brought into main memory upon request.

PA1050 The TOPS-20 Compatibility Package that is merged with any program that uses TOPS-10 UUO's. It simulates the operation of these UUOs by performing the appropriate JSYSs.

Pack A removable set of disks mounted on a common spindle.

Packet A group of words transmitted by means of IPCF to and from jobs and system processes.

Page 1. Any number of lines terminated with a form feed

character. 2. The smallest mappable unit of main memory (512 words). 3. To remove selected parts of a user's program from memory.

Page Fault When the system tries to refer to a page that is not in memory.

Paging The process of transmitting pages of information between main and secondary storage.

Parallel Processes Two processes created by the same superior process.

Parallel processing Concurrent execution of one or more programs.

Parity bit A binary digit appended to a group of bits to make the sum of all the bits always odd or always even.

Parity check A check that tests whether the number of ones or zeros in an array of binary digits is correct. This check helps ensure that the data read has not been unintentionally altered.

PARSER A command language that allows commands to be typed at the console and invokes front-end tasks.

Parsing The systematic ordering of the elements in a command or line of coding. For example, parsing a FORTRAN command places the elements in standard form for the compiler.

Password The character string assigned to a user; it is usually known only to the user and the system. The password is used to verify that a user is entitled to run a job under a specific user name.

Peripheral equipment Any equipment distinct from the CPU that provides the system with outside communication or additional facilities.

Physical address space A set of separate memory locations where information can actually be stored (i.e., main memory) for the purpose of program execution.

PID Process ID. A number assigned to each process by INFO and used by IPCF to route messages between processes.

Pointer 1. A location or register containing an address rather than data. A pointer may be used in indirect addressing or in indexing. 2. An instruction indicating the address, position, and length of a byte of information.

Polling A centrally controlled method of calling a number of points in some sequence to permit them to transmit information.

Priority interrupt An interrupt that usurps control of the computer from the program or monitor and jumps to an interrupt service routine if its priority is higher than the interrupt currently being serviced.

Process An independent task capable of being run by the monitor and having a virtual address space of 512 pages. A process is performed under a job. Also known as a fork or task.

Process identifier An octal number in the range 400001 to 400777 that identifies an inferior process to its superior process.

Program Counter (PC) A register that contains the address from which the next instruction to be executed is fetched.

Programmed operators Instructions that, instead of performing a hardware operation, cause a jump into the monitor system or the user area at a predetermined point and perform a software operation. The monitor (or special user code) interprets these entries as commands from the user program to perform specified operations.

Program trap One of the non-hardwired operation codes that, when decoded by the processor, causes the next instruction to be executed from a specified address.

PROM Programmable Read-Only Memory.

Prompting A system feature that helps users by requesting them to supply arguments necessary to continue processing.

Protection number A number given to every file that defines which users may access the file and what access is to be permitted to them.

PSB Process Storage Block. A table that contains all monitor data for a given process.

Pseudo-operation An instruction to the assembler; an operation code that is not part of the computer's hardware command repertoire.

PTY Pseudo-Teletype. A simulated terminal, controlled by the monitor, on which a subjob runs.

PTYCON Program that controls PTYs.

QUASAR A program that schedules requests for batch jobs and output peripherals such as the line printer.

Queue A list of items waiting to be scheduled or processed according to system, operator, or user-assigned priorities.

RAM Random-Access Memory. Any type of memory with both read and write capability.

Random Access Describes the process of retrieving or depositing information in storage where the time required for such access is independent of the location of the information most recently retrieved or deposited.

Recognition input A method of typing TOPS-20 commands in which the user types the unique portion of a command or argument, and then presses the ESC key. The system attempts to recognize what the user wants to type and either prints its assumption or rings the terminal bell prompting the user to type more. The user can then type more and try for recognition again.

Reentrant program A program consisting of sharable

code that can have several simultaneous independent users.

Register A device capable of storing a specified amount of data, usually one word.

Relative address The specification of a memory location as an offset from a base address. The computer sums the base address and offset to ascertain the relative address. This type of addressing allows the relocation of instructions because the system moves all of the data being addressed by the same amount — the offset.

REL file A file containing one or more relocatable object modules.

Relocatable address An address within a module that is specified as an offset from the first location in that module.

Relocatable code Code written so that it can be located and executed in any part of memory, once it has been linked by the linker.

Relocatable symbols Symbolic names whose associated value is an offset from the base (beginning) of a control section. Such a symbol's value depends on the address of the control section and must be relocated each time the control section is assigned an address by the Linker.

Remote batch A feature that allows data I/O and job control of batch processing from a distant terminal over a communication link.

Remote job entry The input of batch jobs via a cardreader at a remote site and the receipt of job output via a printer or cardpunch at a remote site.

Remote station Unit record and/or terminal equipment, located at a distance from a data processing system with which it communicates via communication lines.

Resident In memory, as opposed to being stored externally.

Resource An available portion of the computer's machinery or software.

RJE Remote Job Entry.

ROM Read-Only Memory. A special memory that can be read from, but not written into.

Root directory Contains the names of all users' directories.

Root segment The segment of an overlay tree that, once loaded, remains in memory during the execution of the task.

Route through The ability of a node in a network to receive data destined for another node and to forward that data directly to its destination or to another intermediate node.

Routine A set of instructions and data for performing one or more specific functions.

RUBOUT A key (also labeled DEL) on the terminal keyboard that erases characters that the user has typed.

RUNOFF A program that facilitates the preparation of typed or printed manuscripts by performing formatting, case shifting, line justification, page numbering, titling, and indexing.

Run-time The time in which a program is executed.

Scheduling algorithm An algorithm that determines the order in which competing jobs are allowed to use resources.

Schema A complete description of a data base.

Secondary storage High-capacity magnetic storage, such as disks.

Sequential access Obtaining data from an I/O device in a serial manner only.

Sharable program A program which can be used by several users at a time.

Shared Pages When the same page is represented in more than one process address space. Allows two or more processes to identify and use the same page of physical storage.

Software Interrupt System Interprets the sequential flow of program execution under a variety of conditions.

Source code Text, usually in the form of an ASCII-formatted file which represents a program. Such a file can be processed by an appropriate system compiler or assembler to produce an object module.

Spooling The technique by which output to slow-speed devices is placed into queues on faster devices (such as disk) to await transmission to the slower devices; this allows more efficient use of the particular device, main memory, and the central processor unit.

SPRINT The batch input processor. SPRINT reads any sequential input stream, sets up the job's control file and data files, and enters the job into the batch input queue.

SPROUT The card punch spooler.

Subcommand A qualifier given with a command that causes the command to operate in a specific way. Contrast with Argument.

Subjob A job created by another job running on the same physical terminal as the job that created it.

Subroutine A routine designed to be used by other routines to accomplish a specific task.

Subscript A numeric valued expression or expression element which is appended to a local or global variable name to uniquely identify specific elements of an array.

Swapping The movement of programs and data between main memory and high-speed secondary storage.

Symbolic address An address used to specify a storage location in the context of a particular program. Symbolic addresses must then be translated into relocatable (or absolute) addresses by the assembler.

Symbolic name Refers to a label used in programs written in source languages to refer to data elements, peripheral devices, instructions, etc.

Symbol table A table containing entries and binary values for each symbol defined or used within a module. This table generally contains additional information about the way in which the symbol was defined in the module.

SYSJOB A program that controls several active system processes simultaneously.

Tape label A record used to identify the magnetic tape on which it resides.

Terminal user Anyone who is eligible to log onto the system.

Thawed Access A type of access that allows a file to be accessed even if other users are reading and writing the file.

Thrashing When the system spends an excessive amount of time swapping pages between memory and the disk.

Throughput The measure of the amount of information a computer can input, process, and output in a given time.

Timesharing 1. A means of sharing the facilities of a computer between several users by giving each one a fraction of a second at a time to execute his program. 2. Pertaining to the *interleaved* use of the time of a device. 3. A method of allocating central processor time and other computer resources to multiple users so that the computer appears to process a number of programs simultaneously; multiprogramming.

Time slice The period of time allocated by the operating system to process a particular partition's program. This term is synonymous with "timesharing interval."

Trap 1. An automatic transfer of control to a prespecified routine that can be caused by software or hardware. 2. A conditional jump to a known location performed automatically by hardware as a side effect of executing a processor

instruction. The address location from which the jump is made is recorded. It is distinguished from an interrupt, which is caused by an external event.

Tree structure A hierarchical structure in which each element may be related to any number of elements at any level below it, but to only one element above it in the hierarchy.

Unit-record device I/O equipment such as a cardreader, cardpunch, or lineprinter.

User name The unique identifier assigned to each user of the system. Also serves as the name of the user's primary directory.

User programs A group of programs, subprograms, or subroutines written by a user.

Variable Any entity that can assume any of a given set of values.

Virtual address A number which indicates relative position within a collection of logically related granules of a storage medium. The fact that the medium itself may be virtual is of little consequence; the ability to deal with a hierarchical or multilevel memory as if it were one medium is one of the principal advantages of systems supporting virtual addressing.

Virtual address space The upper extent of the size of any program the user may create and run.

Virtual memory The space on secondary storage where the monitor keeps a reference copy of the job's image for when the job has to be swapped out of memory to make room for other jobs.

Virtual memory pointer Pointer used to keep track of the program segments in memory.

Wait state The condition of a process that is dependent on one or more events in order to enter the active state. Same as a blocked state.

Word An ordered set of bits that occupies one storage location and is treated by the computer as a unit.

Working set The set of pages most recently accessed by a process. The working set is modified as the process runs.

Write-lock condition The condition of a device that is protected against any transfers which would write information to it.

Index

A

- Accessories and Supplies Group (A&SG), 11-4
- accounting facility, 3-3
- addressing, virtual memory, 4-4
- address space
 - in KL10-E processor, 5-7
 - in KS10 processor, 6-3
- ALGOL, 3-1, 8-4—8-5
- ALGOL-20, 8-4—8-5
- ALGOTS (ALGOL object-time system), 8-5
- APL, 3-1, 8-7—8-9
- APL-SF, 8-7—8-9
- application programmers, 3-1
- applications, 2-1, 9-1—9-4
- application tools, 3-1
- architecture
 - DNA, 10-1—10-2
 - KL10-E memory subsystem in, 5-5
 - of KL10E processor, 5-1
 - of KS10 processor, 6-1—6-3
- archiving files, 4-6
- archiving/migration feature, 3-3
- arithmetic
 - using BASIC-PLUS-2, 8-7
 - fixed- and floating-point, 5-3, 6-2
- arithmetic testing instructions, 5-3, 6-2
- ARPANET (Advanced Research Project Agency Network), 2-3
- assembler language, 3-1, 8-1
- assignment statements, 8-8
- authorization of users, 3-2
- automatic restarts, 2-3, 3-3
- Automatic Volume Recognition (AVR), 4-6
- auxiliary processors, 7-6
- availability, 2-3

B

- backup, 3-3
- balance sets, 4-2—4-4
- BASIC-PLUS-2, 3-1, 8-5—8-7
- Basic Service, 11-3
- batch processing, 2-2, 4-1—4-2
 - by application programmers, 3-1
 - in networks, 10-2
- batch streams, 3-3
- bias control feature, 3-3
- Binary Synchronous protocol (BISYNC; BSC), 4-10
- BLISS-36, 3-1—3-2, 8-9—8-10
- BLOCK-IN/BLOCK-OUT instruction, 5-4
- block structure in ALGOL, 8-4
- block transfer instructions, 5-4
- bootstrapping reliability and, 2-3
- branch statements, 8-8
- buffers
 - in APL, 8-9
 - in paper tape reader/punch, 7-4
 - print line, 7-2, 7-3
- buses
 - in KL10-E processor, 5-1, 5-8—5-10
 - KS10, 6-1—6-2
- Business Instruction Set, 5-4, 6-3

- byte manipulation, 5-3, 6-2
- byte transfers, 5-10

C

- cache memories, 2-1
 - in KL10-E processor, 5-5—5-7
 - in KS10 processor, 6-2
- CALL statement, 8-7
- cardreaders, 7-4
- CASE statement, 8-5
- CD20 cardreaders, 7-4
- centralized queue manager, 4-1—4-2
- central processing units — see *processors*
- CHAIN statement, 8-7
- CHANGE statement, 8-6
- channel bus (Cbus), 5-9
- Charaband, 7-3
- characters for APL, 8-8
- class scheduling, 3-2—3-3
- COBDDT (COBOL debugger), 8-3
- COBOL, 3-1, 8-2—8-4
- COBOL-68, 8-2, 8-4
- COBOL-74, 8-2, 8-4
- CODASYL, 9-1
- COGO-20 utility, 9-4
- command language, 2-1—2-2
- Command Language Processor, 2-3, 4-1
 - invoked by jobs, 4-2
- command parser, 5-8
- Command Recognition, 2-2
- commercial applications, 2-1
- communications, 2-1
 - console functions for, 4-7
 - interfaces for, 7-6
 - KL10-E I/O subsystem and, 5-8
 - by processes, 4-2—4-3
 - software for, 2-2—2-3, 10-1—10-5
- Computer Special Systems (CSS), 11-3—11-4
- Computer Supplies group, 11-4
- conditional sales agreements, 11-4
- conditional statements, 8-7
- console front-end processors, 4-6—4-7
 - KL10-E, 5-7—5-8
- console mode, 6-4—6-5
- console subsystems, 2-1, 4-7
 - KL10-E front-end subsystem, 5-7—5-8
 - for KS10 processor, 6-4—6-5
- console terminals (CTYs), 6-4—6-5
- Control RAM Address (CRA) Module, 6-1
- Control RAM Memory (CRM) Module, 6-1
- Conventional Format, 8-4
- conversion packages for APL, 8-9
- copy-on-write feature, 2-2, 4-5
- Core Status Table, 6-4
- courses, 11-1—11-3
- CPL (Conversational Programming Language), 3-1, 8-10
- CPUs — see *processors*
- CREF program, 3-3
- CTYs (console terminals), 6-4—6-5
- customer financing, 11-4
- Customer Spares, 11-4—11-5
- cyclic redundancy checking (CRC), 7-1

D

- data
 - in APL, 8-8
 - in BASIC-PLUS-2, 8-6—8-7
 - management of, 9-1—9-4
 - sharing of, 4-6
 - types of in COBOL, 8-3
- Data Access Protocol (DAP), 10-2
- Database Control System (DBCS), 9-2
- database management, 3-1, 9-1—9-2
- Data Definition Language (DDL), 9-1
- data dictionary, 9-3
- Data Manipulation Language (DML), 9-1—9-2
- Data Path Executive (DPE Module), 6-1
- Data Path Memory (DPM) Module, 6-1
- DAVFUs (direct-access vertical form units), 7-3
- DBMS (database management system), 3-1, 9-1—9-3
- DBMS-20, 9-1—9-2
- DDCMP (DIGITAL Data Communications Message Protocol), 10-2
- DDT (DIGITAL Debugging Tool), 3-4, 8-1
- debugging
 - in APL, 8-9
 - in BASIC-PLUS-2, 8-7
 - in BLISS-36, 8-9
 - in COBOL, 8-3
 - in CPL, 9-10
 - DDT for, 8-1
 - in FORTRAN, 8-2
- DECnet-20, 2-2, 10-1—10-3
- DECservice, 11-3
- DECSYSTEM-20 ALGOL, 3-1
- DECSYSTEM-20 COBOL, 3-1
- DECSYSTEM-20 FORTRAN, 3-1
- DECSYSTEM-2020
 - KS10 processor and, 6-1—6-5 (see also *KS10 processor; KS systems*)
- DECUS (Digital Equipment Computer Users Society), 11-5
- deferred mode, 9-2—9-3
- devices — see *hardware; peripherals*
- DH11 programmable asynchronous serial line multiplexers, 7-6
- diagnostics
 - console and, 4-7
 - in KL10-E processors, 5-1, 5-7
 - KLINIK for, 2-1
 - in KS10 processor, 6-4
 - online, 2-3
- Digital Data Communications Message Protocol (DDCMP), 10-2
- Digital Equipment Computer Users Society (DECUS), 11-5
- DIGITAL Network Architecture (DNA), 10-1—10-2
- Digital ten-to-eleven (DTE-20) interface, 5-7, 5-10
- direct-access vertical form units (DAVFUs), 7-3
- directories
 - in caches, 5-6
 - user file, 4-6
- direct process control, 4-2
- disk controllers, 7-1

disk I/O statistics, 3-3
disks, 7-1
 RP06 drives, 5-8
 storage quotas on, 3-2
DL11 serial line asynchronous interfaces, 7-6
DN20 communications front end, 7-6, 10-5
DNA (DIGITAL Network Architecture),
 10-1—10-2
double-precision floating-point format, 5-3
DTE interface, 5-2, 5-7, 5-10, 7-6
DUMPER program, 3-4
DUP11 single synchronous line interfaces, 7-6
DX/TOPS-20 document transmission, 10-5
DZ11 terminal line interfaces, 7-6

E
EBox (Execution Box), 5-1—5-4
EBR register, 4-5
EDIT program, 3-4
Educational Services, 11-1—11-3
enqueue/dequeue (ENQ/DEQ) facility, 4-3
ENTER facility, 8-3
errors
 analysis of and recovery from in APL, 8-9
 in batch processing, 4-1
 corrections for in tape systems, 7-2
 detection and reporting of, 2-3
 handling of in BASIC-PLUS-2, 8-7
 interrupt processing for, 2-1
 journals for recovery from, 9-2
 in MOS memory, 5-5, 6-4
Execution Box (EBOX), 5-1—5-4
execution mode, 8-8
executive mode, 5-2, 6-3

F
fast register blocks, 5-4
Field Service, 11-1, 11-3
FILCOM program, 3-3
file directories, 4-6
file-handling services, 3-1
files, 4-5
 in APL, 8-9
 in BASIC-PLUS-2, 8-6
 in COBOL, 8-3
 for data bases, 9-1
 in DBMS, 9-3
 in networks, 10-2
file system, 2-2, 4-5—4-6
financing, 11-4
fixed/floating conversion instructions, 5-3, 6-2
fixed-point arithmetic, 5-3, 6-2
flags for MOS memory errors, 5-5
floating point arithmetic, 5-3
FORDDT debugger, 8-2
forms, TRAFFIC-20 for, 3-1, 9-4
FOROTS (FORTRAN object-time system), 8-1,
 8-2
FORTRAN, 3-1, 8-1—8-2
FORTRAN IV, 8-2
FORTRAN-208-1—8-2, front-end
 subsystems, 2-1, 4-7
 DN20 communications front end, 7-6, 10-5
 in KL10-E processor, 5-7—5-8
full-word data transmission, 5-3, 6-2
function-definition mode, 8-8

G
GALAXY system, 10-4, 10-5
GCR mode, 7-2

general purpose registers, 5-4, 6-2
geometry COGO-20 for, 9-4
groups, 4-6

H
half-word data transmission, 5-3, 6-2
hardcopy terminals, 7-4—7-5
hardware
 installation of, 11-1
 KL10-E processors, 5-1—5-10
 KS10 processors, 6-1—6-2
 mass storage peripherals, 7-1—7-2
 reliability of, 2-3
 support services for, 11-3
 terminals and interfaces, 7-4—7-6
 unit record peripherals, 7-2—7-4
Hardware Page Table, 4-5
HASP protocol emulators, 10-4
help facilities, 2-2

I
IBM Emulation/Termination, 2-2
IBM systems, Internets for, 10-4
immediate mode, 8-7, 8-8, 8-10
Immediate Mode Pointer, 4-5
indexed access method, 4-4
indexed sequential-access mode (ISAM), 8-3
indirect pointers, 6-4
inferior processes, 4-2
input queues, 4-2
input spoolers, 4-1
installation, 11-1
instructions and instruction sets
 for KL10-E processor, 5-1—5-4
 for KS10 processor, 6-2—6-3
interactive mode, 9-2
interactive processing
 by application programmers, 3-1
 in COBOL, 8-3
Interactive Query Language (IQL), 9-2—9-4
interfaces, 7-6
 DTE, 5-2, 5-7, 5-10
 for NETCON, 10-3
 USART, 6-4
internal memory in KL10-E processor, 5-5
Internets, 10-3—10-5
Interprocess Communications Facility (IPCF),
 4-2
interrupts, 2-1
 priority system for, 5-4
 software interrupt system for, 4-2
interval timer, 5-4
I/O instructions, 5-3, 5-4
I/O subsystems in KL10-E processor, 5-1,
 5-8—5-10
IPCF (Interprocess Communication Facility),
 4-2
IQL (Interactive Query Language), 9-2—9-4
ISAM (indexed sequential access mode), 8-3

J
jobs, 4-2
 batch processing of, 4-1
 job statistics, 3-3
 journals, 9-2
 JSYS instruction, 5-4

K
KL10 processor, 2-1
KL10-E processor, 5-1—5-10
 fixed-point arithmetic on, 6-2

KLINIK (diagnostics), 2-1, 5-7—5-8
KLINIT (initialization program), 5-8
KL systems
 DH11 proramable asynchronous serial
 line multiplexers for, 7-6
 line interfaces supported on, 7-4
KMC11-A auxiliary processors, 7-6
KS10 bus, 6-1—6-2
KS10 processor, 2-1, 6-1—6-5
 peripherals supported on, 7-1
KS systems
 interfaces for, 7-6
 line interfaces supported on, 7-4

L
LA38 DECwriter IV hardcopy terminals, 7-5
LA120 hardcopy terminals, 7-4—7-5
labeling of tapes, 4-6
languages, 2-1, 2-2, 3-1
 APL, 8-7—8-9
 BASIC-PLUS-2, 8-5—8-7
 COBOL, 8-2—8-5
 CPL, 8-10
 FORTRAN, 8-1—8-2
leasing, 11-4
library facilities in COBOL, 8-3
lineprinters, 7-2—7-4
 output spooling for, 4-2
LINK program, 3-4
log files, 4-1
logical communications links, 10-2
logical testing instructions, 5-3, 6-2—6-3
logic instructions, 5-3, 6-2
longitudinal redundancy checking (LRC), 7-1
LP05 lineprinters, 7-2—7-3
LP07 lineprinters, 7-3
LP14 lineprinters, 7-3
LP20 lineprinter controllers, 7-2, 7-3
LP20-A and -B lineprinters, 7-2—7-3
LP20-C and -D lineprinters, 7-3
LP200 lineprinters, 7-3—7-4

M
MACOR (assembly language), 3-1, 8-1
MAIL program, 3-3
maintenance
 console and, 4-7
 of hardware, 11-3
 for LA38 hardcopy terminals, 7-5
 of software, 2-3, 11-1
MAKLIB program, 3-3
mapping
 in KL10-E processor, 5-7
 in KS10 processor, 6-3—6-4
MASSBUS in KL10-E processor, 5-1, 5-9
mass storage — see *memory*
matrix manipulation, 8-7
MBox (memory control unit), 5-1, 5-5, 5-6
memory, 2-1
 disk storage quotas for, 3-2
 mass peripherals for, 7-1—7-2
 organization of, 2-2
 sharing of, 4-3
 virtual, 4-4—4-5, 8-6
memory control unit (MBox), 5-1, 5-5, 5-6
memory mapping, 5-7, 6-3—6-4
memory status table, 4-5
memory subsystems
 in KL10-E processor, 5-5—5-7
 in KS10 processor, 6-3—6-4

messages MAIL program for, 3-3
meters, 5-4
microprogramming, 6-1
microstore, 6-2
modes
 for APL language statements, 8-8
 DTE-20 operating, 5-10
 processor, 5-2, 6-3
modification instructions, 5-3, 6-2—6-3
monitor, 4-3—4-4
monitor calls, 3-2
Monitor Page Table, 4-5
monitor statistics, 3-3
MOS (Metal Oxide Semiconductor) memory,
 5-5, 6-4
MS10 memory, 6-4
multiplexed I/O bus, 5-8
multiprogramming environments, system
 programming for, 3-1

N

NETCON (Network Control Program) utility,
 10-1, 10-3
network file transfer (NFT) utility, 10-2, 10-3
networks
 DECnet-20, 10-1—10-3
 DN20 communications front end for, 7-6
 Internets, 10-3—10-5
Network Services Protocol (NSP), 10-2
NRZI mode, 7-2
numbers fixed- and floating-point, 5-3, 6-2

O

object-time systems
 ALGOTS, 8-5
 FOROTS, 8-1, 8-2
offset mode, 5-4
ON ERROR GOTO statement, 8-7
operating system — see *TOPS-20*
operators — see *system operators*
OPR unified system interface, 3-3
optimization in FORTRAN-20, 8-2
output spoolers, 4-2
OWN variables (ALGOL), 8-5

P

pages, 4-4—4-5
 in caches, 5-6
 in KL10-E memory mapping, 5-7
 in KS10 memory mapping, 6-3
page tables, 5-7
paper tape reader/punches, 7-4
parameters in ALGOL, 8-5
parity, 7-1
passwords, 3-2
PC20 paper tape reader/punches, 7-4
PCS-20 (Project Control System), 9-4
PDP-11 subsystem, 5-1, 5-7, 5-8, 5-10
PE mode, 7-2
Performance Analysis Statistics program, 3-3
peripherals, 2-1
 console interface with, 4-7
 KL10-E I/O subsystem and, 5-8
 MASSBUS for, 5-9
 mass storage, 7-1—7-2
 supported by KL10-E front-end
 subsystem, 5-7
 terminals and interfaces, 7-4—7-6
 unit record, 7-2—7-4

physical address space
 in KL10-E processor, 5-7
 in KS10 processor, 6-3
physical communications links, 10-2
PID (Process Identification Number), 4-2
PL/I, CPL and, 8-10
PLEASE program, 3-3
pointers, 4-4, 4-5, 6-3—6-4
postpurging, 4-4
power failures, automatic recovery after, 3-3
preloading, 4-4
printers, 7-2—7-4
 hardcopy terminals, 7-4—7-5
priorities
 for interrupts, 5-4
 for processes, 4-2
private pointers, 6-3
privileged mode, 5-10
privileges, 3-2
procedures in ALGOL, 8-4—8-5
processes, 4-2
 communications by, 4-2—4-3
 memory organization and, 2-2
 virtual memory and, 4-4
Process Identification Number (PID), 4-2
processor modes, 5-2, 6-3
processors, 2-1
 auxiliary, 7-6
 console front-end processors, 4-6—4-7
 KL10-E, 5-1—5-10
 KS10, 6-1—6-5
 scheduling time on, 3-2—3-3
professional services, 11-1
program control instructions, 5-4, 6-3
programmable address break, 5-4
programmable asynchronous serial line
 multiplexers, 7-6
programmable vertical format units (PVUFs),
 7-2, 7-3
programs
 in COBOL, 8-3—8-4
 in CPL, 8-10
 sharing of, 4-6
program segmentation, 8-7
protection for files, 4-5—4-6
protocol emulators, 10-3—10-5
pseudoterminals, 7-4

Q

quadwords, 5-6
? (question mark), 2-2
queue manager, 4-1—4-2

R

RDMAIL program, 3-3
record-handling services, 3-1
record input/output in BASIC-PLUS-2, 8-6
records in DBMS, 9-1, 9-2
recovery, 2-3, 3-3
registers
 in KL10-E processor, 5-4
 in KS10 processor, 6-2
 for virtual memory, 4-5
reliability, 2-3
remote job entry stations, 10-3
reports, IQL for, 9-3—9-4
RERUN facility, 8-4
resources
 accounting statistics for, 3-3
 system manager control of, 3-2

restricted mode, 5-10
RH20 integrated controllers, 5-9
RM03 disk-pack subsystems, 7-1
ROOT-DIRECTORY, 4-6
RP06 disk-pack subsystems, 5-8, 7-1
RSX software, 5-8
RTP20 disk subsystems, 7-1

S

Scheduler, 4-4
scheduling
 centralized queue manager for, 4-1—4-2
 system manager controls on, 3-2—3-3
 TOPS-20 monitor for, 4-3
SCHEMA, 9-1, 9-2
scientific applications, 2-1
security, 2-3
 during diagnostics, 4-7
 user authorization for, 3-2
Self-Maintenance Service for Software, 11-1
self-test modules, 7-2, 7-3
serial line asynchronous interfaces, 7-6
serial line multiplexers, 7-6
services, 11-1—11-5
Shared Mode Pointer, 4-5
Shared Page Table (SPT), 4-5, 6-3
shared pointers, 6-3
sharing
 file system for, 4-6
 of memory, 4-3
single-precision floating-point format, 5-3
single synchronous line interfaces, 7-6
skip instructions, 5-3, 6-2—6-3
SLEEP statement, 8-7
software
 communications, 10-1—10-5
 for data administration, 9-1—9-4
 process structure of, 2-2
 reliability of, 2-3
 RSX, 5-8
 support services for, 11-1 (see also *TOPS-20*)
software interrupt system, 4-2, 4-3
Software Product Updates, 11-1
SORT/MERGE utility, 9-4
spoolers
 input, 4-1
 output, 4-2
SPT (Shared Page Table), 4-5, 6-3
SPT index, 6-4
statements
 in APL, 8-8
 in IQL, 9-3
statistics, 3-3
 Command Language Processor and, 4-1
STOP statement, 8-7
storage — see *memory*
STORE statement, 9-2
string arrays, 8-6
string constants, 8-5
STRING instructions, 5-4, 6-3
string manipulation, 8-3
string variables, 8-6
subdirectories, 4-6
Sub-Schema, 9-1
superior processes, 4-2
supplies, 11-4
support services, 11-1—11-5

- swapping, 4-4
- switches
 - in ALGOL, 8-5
 - in batch system, 4-1
- system managers, 3-2—3-3
- system operators, 3-3
 - for batch processing, 4-1
- system programmers, 3-1—3-2
- system recovery, 2-3, 3-3
- system utilization statistics, 3-3

T

- tapes
 - labelling of, 4-6
 - peripherals for, 7-1—7-2
- task-to-task communications, 10-1, 10-2
- terminal drivers, 7-4
- Terminal Format, 8-4
- terminal line interfaces, 7-6
- terminals, 7-4—7-6
 - for APL, 8-8
 - consoles and, 4-7, 6-4
 - in networks, 10-2
 - remote job entry stations as, 10-3
- time base counter, 5-4
- TOPS-20 (operating system), 2-1—2-3, 4-1—4-7
 - in DECnet configurations, 10-1
 - Educational Services courses on, 11-2—11-3

- Internets supported on, 10-3—10-5
- memory assigned by, 5-7
- NETCON commands for, 10-3
- utility programs with, 3-3—3-4
- TOPS-20AN (communications software), 2-3
- TOPS-20 assembler language, 8-1
- TOPS-20 monitor, 4-3—4-4
- TRAFFIC-20 (Transaction Routing and Forms Formatting in COBOL), 3-1, 9-4
- transaction processing, 3-1
- trap handling, 5-4, 6-3
- TU70 series magnetic tape systems, 7-1—7-2
- TU77 magnetic tape systems, 7-1, 7-2
- TV program, 3-4

U

- UBR register, 4-5
- UNIBUS
 - in KL10-E processor, 5-1, 5-7, 5-9—5-10
 - peripherals attached to, 7-1
- UNIBUS Adapters (UBAs), 2-1, 7-1
- unimplemented user operations(UUOs), 5-4
- unit record peripherals, 7-1—7-4
- universal synchronous/asynchronous receiver/transmitter (USART)
 - interfaces, 6-4
- UPDATE feature, 8-6
- user authorization, 3-2

- user file directory, 4-6
- user mode, 5-2, 6-3—6-5
- User Page Table, 4-5
- users, 3-1—3-4
 - classes of, for file protection, 4-5
- user utilities, 3-3—3-4
- utilities, 3-3—3-4, 9-4
 - with DBMS, 9-2
 - NETCON, 10-1, 10-3

V

- video terminals, 7-5—7-6
- virtual memory, 4-4—4-5
- virtual memory arrays, 8-6
- VT100 video terminals, 7-5—7-6

W

- WAIT statement, 8-7
- warranties, for software, 11-1
- word processing, DX/TOPS-20 document
 - transmission on, 10-5
- working set manager, 4-4
- working sets, 4-2, 4-4
- workspaces, in APL, 8-9
- WPS-8 word processing systems, 10-5

Z

- zero, 5-3, 6-2

digital

DIGITAL EQUIPMENT CORPORATION, Corporate Headquarters: Maynard, MA 01754, Tel. (617) 897-5111 — SALES AND SERVICE OFFICES: UNITED STATES — ALABAMA, Birmingham, Huntsville ARIZONA, Phoenix, Tucson ARKANSAS, Little Rock CALIFORNIA, Costa Mesa, El Segundo, Los Angeles, Oakland, Sacramento, San Diego, San Francisco, Monrovia, Santa Barbara, Santa Clara, Sherman Oaks COLORADO, Colorado Springs, Denver CONNECTICUT, Fairfield, Meriden DELAWARE, Newark FLORIDA, Miami, Orlando, Pensacola, Tampa GEORGIA, Atlanta HAWAII, Honolulu IDAHO, Boise ILLINOIS, Chicago, Peoria INDIANA, Indianapolis IOWA, Bettendorf KENTUCKY, Louisville LOUISIANA, New Orleans MARYLAND, Baltimore MASSACHUSETTS, Boston, Springfield, Waltham MICHIGAN, Detroit, Kalamazoo MINNESOTA, Minneapolis MISSOURI, Kansas City, St. Louis NEBRASKA, Omaha NEW HAMPSHIRE, Manchester NEW JERSEY, Cherry Hill, Parsippany, Princeton, Somerset NEW MEXICO, Albuquerque, Los Alamos NEW YORK, Albany, Buffalo, Long Island, New York City, Rochester, Syracuse, Westchester NORTH CAROLINA, Chapel Hill, Charlotte OHIO, Cincinnati, Cleveland, Columbus, Dayton OKLAHOMA, Tulsa OREGON, Portland PENNSYLVANIA, Harrisburg, Philadelphia, Pittsburgh RHODE ISLAND, Providence SOUTH CAROLINA, Columbia, Greenville TENNESSEE, Knoxville, Nashville TEXAS, Austin, Dallas, El Paso, Houston, San Antonio UTAH, Salt Lake City VERMONT, Burlington VIRGINIA, Fairfax, Richmond WASHINGTON, Seattle, Spokane WASHINGTON D.C. WEST VIRGINIA, Charleston WISCONSIN, Milwaukee INTERNATIONAL — EUROPEAN AREA HEADQUARTERS: Geneva, Tel: [41] (22)-93-33-11 INTERNATIONAL AREA HEADQUARTERS: Acton, MA 01754, U.S.A., Tel: (617) 263-6000 AUSTRALIA, Adelaide, Brisbane, Canberra, Hobart, Melbourne, Perth, Sydney, Townsville AUSTRIA, Vienna BELGIUM, Brussels BRAZIL, Rio de Janeiro, Sao Paulo CANADA, Calgary, Edmonton, Halifax, Kingston, London, Montreal, Ottawa, Quebec City, Regina, Toronto, Vancouver, Victoria, Winnipeg DENMARK, Copenhagen ENGLAND, Basingstoke, Birmingham, Bristol, Ealing, Epsom, Leeds, Leicester, London, Manchester, Reading, Welwyn FINLAND, Helsinki FRANCE, Bordeaux, Lyon, Paris, Puteaux, Strasbourg HOLLAND, Amstelveen, Delft, Utrecht HONG KONG IRELAND, Dublin ISRAEL, Tel Aviv ITALY, Milan, Rome, Turin JAPAN, Osaka, Tokyo MEXICO, Mexico City, Monterrey NEW ZEALAND, Auckland, Christchurch, Wellington NORTHERN IRELAND, Belfast NORWAY, Oslo, PUERTO RICO, San Juan SCOTLAND, Edinburgh REPUBLIC OF SINGAPORE SPAIN, Barcelona, Madrid SWEDEN, Gothenburg, Stockholm SWITZERLAND, Geneva, Zurich WEST GERMANY, Berlin, Cologne, Frankfurt, Hamburg, Hannover, Munich, Nurnberg, Stuttgart