TOPS-10/20 SALES TRAINING
STUDENT GUIDE

<<For Internal Use Only>>

KL ARCHITECTURE

The current architecture of the KL Central Processing Unit (CPU) evolved from a long tradition in designing and implementing "interactive" computing systems. Digital's first 36-bit computer system, the PDP-6, appeared in 1964. The PDP-6 was designed for both a timesharing computational environment and real-time laboratory use with direct interfacing capability. The system was constructed of germanium and silicon transistors, with modules plugged into soldered connectors.

In 1967, the KA10 CPU replaced the PDP-6. The KA used Digital's popular Flip Chip modules for logic implementation. Wire wrap technology was used for backplane wiring.

In 1972, the KI10 CPU arrived. The KI used TTL logic and introduced "associative memory hardware", which ultimately led to virtual memory implementation.

In 1975, the KL processor (marketing designation - "DECSYSTEM-2040") was introduced to run the TOPS-10 and TOPS-20 operating systems. The KL was implemented in ECL (Emitter-Coupled Logic), and included a sophisticated cache memory design plus expanded memory hardware. The KL was designed as a microprogrammed machine; instructions were to be implemented through microcode rather than "hard" wiring. It featured the addition of a front-end processor and I/O channels (for disks, tapes, and communications) integrated into the CPU design.

At the time the KL was designed, Digital had significant experience in "multi-task" (often called "timesharing") environments. TOPS-10 was well understood; TOPS-20 was also well understood, since it evolved from TENEX (written for the ARPA net). An important goal of the KL's architecture and implementation design was hardware support of (1) many functions previously accomplished by software, and (2) functions that were difficult or impossible to accomplish through software without undue overhead. Hardware and software architects worked closely together to produce this machine. Many important KL features resulted from their conception of the environment the KL would support, prior to the machine's actual design.

A primary design goal built into each generation of Digital's 36-bit family (PDP-6, KA, KI, KL, KS) is software compatibility at the assembly language level. Each succeeding processor is capable of running both user and systems programs from the previous family member. The reverse (running software developed for newer processors on previous family members) is not necessarily possible, due to added features in each generation of machines. For example, the TOPS-1Ø Monitor, Release 7.Ø1, does not support KA processors.

The concept of compatibility between one generation and the next underlines a key element in the KL product strategy: protecting the software investment of our customers and ourselves. This philosophy allows the installed CPU to produce an excellent revenue base over the years, through growth and upgrades in processor models. Hence, the KI is a "drop-in" replacement for the KA, the KL is a "drop-in" replacement for the KA and the KI.

Each family member exhibits architectural, logic and performance improvements over its predecessor. One example is the measure of instructions per second. The KA "average" rating is 38Ø,ØØØ instructions per second, or Ø.38 mips (million instructions per second). The KI rates approximately Ø.72 mips, and the KL rates 1.8 mips.

The selling price of the three processors can be considered essentially the same, if the value of a purchase dollar is considered. Therefore, the basic themes of the 36-bit processor evolution are: to take advantage of contemporary technology in unit design and to provide significant price/performance advantages over the previous model (same price/higher performance). For example, the KL displays a performance improvement factor of almost five over the KA, for programs using the basic instruction set. An even larger performance improvement is realized for COBOL or extended precision scientific programs, since the KL has an "Extended Instruction Set" and double-precision hardware, and the KA does not.

THE E-BOX

The E-Box, or Execution Box, is that element of the KL processor which executes instructions and performs various control functions for overall CPU operation. The KL supports 398 microprogrammed instructions. Each instruction is a series of microinstructions performing various logical functions, i.e.: processor state control, data path control, and the implementation of each instruction. The use of microprogramming in the KL provides several important benefits:

- The CPU can execute either TOPS-1Ø or TOPS-2Ø paging and memory management with the same hardware. Only the microstore must be changed.
- The CPU can support either the TOPS-1Ø (using UUOs) or TOPS-2Ø (using JSYS calls) Monitor Call interfaces.
- Engineering changes or CPU improvements can often be implemented by microstore (firmware) changes, rather than by wiring changes.
- The packaging technology of microcoded hardware facilitates small size and flexibility, relative to other technologies.
- Cache and virtual memory operations are enhanced by using microcode.
- Architectural extensions can often be made using microprogramming. An important example is in the support of new or advanced peripherals subsystems.

The microstore is actually implemented through the use of 2,Ø48 (2K) 75-bit words.

The E-Box also contains eight (8) sets of sixteen general purpose registers, four timing and accounting meters, and an interface to the low-speed I/O devices (DTE for UNIBUS equipment, and DIA/DIB for multiplexed I/O bus devices). The interface to these low-speed control devices is controlled by the E-Box microcode and operates by stealing microcode cycles between the execution of CPU instructions.

Key elements in the KL's ability to perform successfully in a multi-user environment are the two processor modes: USER mode and EXECUTIVE mode. Instructions are always executed in one of these two modes. This concept (processor modes) has been the foundation of Digital's timesharing for over a decade.

In a multi-task environment, a user must be prevented from executing certain instructions that might jeopardize other user programs. Under KL implementation, an attempt by a user to execute an improper instruction results in a "trap" to the Monitor. The Monitor then analyzes the user instruction and takes appropriate action. Improper instructions include I/O instructions (since peripheral devices are shared by many users), attempts to change the priority interrupt system, and attempts to execute a processor "HALT". This "trapping" action occurs whenever the processor is in USER mode and such an instruction is encountered.

In EXECUTIVE Mode, all implemented instructions are legal. The Monitor operates in EXECUTIVE (or EXEC) mode and is able to control all system resources and the state of the processor. Table 1 further illustrates this concept.

Table 1. Processor Modes

User Mode

Public Submode
- User programs.
- 256K word address.
- Permits all instructions, 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.

Table 1. (Continued)
Executive Mode
(Monitor)

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

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

The KL has an impression instruction set that is logically complete and easy to learn. The instruction set itself is probably beyond any discussion you may have with a prospect; however, more detailed information is included in the KL Technical Summary, available in all sales offices. In addition to this document, a myriad of information on the KL instruction set is available from your local Software Specialist, for sales purposes.

Eight sets of 16 general purpose registers (also known as accumulators) are provided in the KL architecture. They can be used as:

- Accumulators
- Index Registers
- The first 16 locations of main memory.

Since register addressing bits (9-12 and/or 14-17) are 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, thus eliminating the need for storing register contents when switching from USER mode to EXECUTIVE mode.

Four meters are built in to the KL to provide several timing and counting functions. These include:

- An Interval Timer, which provides a programmable source of interrupts with a one microsecond resolution. It is used for real-time applications (under TOPS-10) and for page management.

- The Time Base, which provides a 60-bit, one microsecond resolution counter to generate a source of elapsed time.

- The Accounting Meters, which provide an accurate measurement of the amount of processor resource used by a job.

- The Performance Analysis Counter, which provides a tool for studying hardware and software performance. It may be used to measure such events as cache hit rate, interrupts, and channel activity.

THE MEMORY SUBSYSTEM

A significant aspect of the KL's architectural implementation is the control of all memory via a discrete memory controller, called the M-Box. The M-Box is responsible for all memory requests from both the E-Box and the integrated channel/controller hardware, which interfaces with high-speed peripherals (disks and tapes). You recall that slow-speed peripherals present their requests to the E-Box (through a microcode implementation), and are in turn presented by the E-Box to the M-Box for actual memory access. In addition, the M-Box performs the important paging (address translation) for memory management. The M-Box contains the cache memory, for processor models so equipped.

The M-Box supports either external or internal memory. Figure 3-1 illustrates a KL processor equipped with external core memory. External memory support requires the use of a DMA (Direct Memory Access) controller to drive the long memory bus lines. Internal memory support does not require this DMA controller. Figure 3-2 illustrates a KL equipped with internal core or MOS memory.

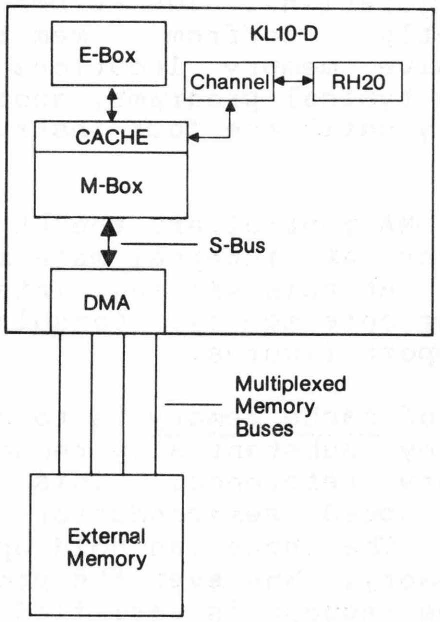


FIG. 3-1. EXTERNAL MEMORY ARCHITECTURE

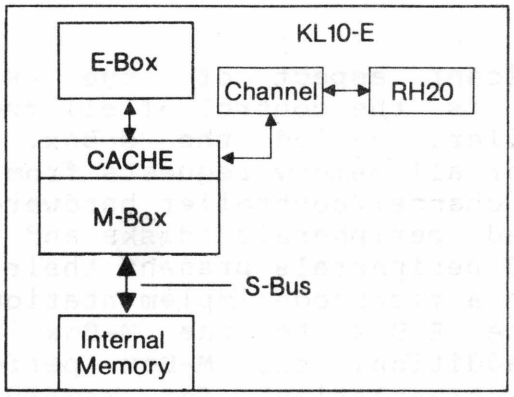


FIG. 3-2. INTERNAL MEMORY ARCHITECTURE

To meet the requirements of large systems operating the TOPS-1Ø Monitor, the KL supports up to 16 external core memory modules that provide a maximum of 4,Ø96K direct-address, 36-bit words. The DMA supports four memory buses, allowing interleaving and multiple-word fetches from separate memory modules. The concept of interleaving is a well-established method to increase the effective bandwidth of memory configurations. The KL/DMA reads or writes four words concurrently to/from memory. These four logically-successive memory locations are placed in the cache memory. In typical programs, another memory reference is not necessary until the four instructions or data words are processed.

Without the DMA controller, the KL can support up to 1.5 million words of internal main memory (a figure that grows each year). At this writing, internal main memory can be MOS memory or core memory. Consult pricing summary for latest memory support figures.

The purpose of cache memory is to decrease instruction execution time by substantially reducing the average time needed for a memory reference. This is accomplished by placing a high speed semiconductor memory (the "cache") inside the M-Box. The cache can hold up to 2K words from core/MOS/main memory. Whenever the program accesses a word held in cache, the request is satisfied in 133 nanoseconds, as opposed to 867 nanoseconds or more for a main memory reference (as measured at the E-Box).

The KL cache design is extremely sophisticated, with an effective capability that is considered "state-of-the-art". The hit rate of the KL cache memory usually exceeds 93%. There are several unique design features of the KL cache that deserve attention:

- KL cache is not a write-through cache. Therefore, time is not wasted in writing through to main memory (in a tight compute loop with an index, for example).

- Hardware is provided to manage the words contained in cache memory. Cached locations change constantly with varying system demands, but the cache algorithm is based on the assumption that memory references tend to be localized.

- Memory is updated only when a word in cache must be emptied to make room for another. The hardware takes care of this housekeeping. It is totally transparent to the software, at all times.

An excellent discussion of cache memory is included in the current version of the KL Technical Summary, available in all local sales offices.

The KL uses hardware to implement memory address mapping, from a program's memory address space (called the effective address or the virtual address) to the actual or physical memory address space in the system's main memory. This mapping is accomplished by substituting hardware logic for certain bits in the memory address. Memory mapping implementation provides the following benefits:

- Physical memory can be up to 16 times larger than a user's virtual address space. This is especially important in a multi-user system, since many users may concurrently reside in main memory. Each user thinks he has a physical memory of 256K words, beginning with location "00000". In reality, the Monitor places this user's address space in the optimal location in the larger main memory.

- When the Monitor assigns a user's physical memory area (for the quantum of time the user's program is to run), the hardware prevents the user from addressing memory outside of this physical area. Thus, one user can not interfere with another - even if he makes obvious mistakes in his program, relative to addressing. If a user tries to access memory outside the assigned address space, a hardware error trap allows the Monitor to notify the user of the error. Under no circumstances is the access allowed. If a user is not utilizing all of the assigned 256K user address space and his/her program expands (as is the case in compiles and editing operations), the Monitor can easily assign additional memory space.

- Memory mapping takes place on a "page" basis; a user is given a certain number of 512-word pages, depending upon the amount of memory space required. This memory allocation can be "scattered" throughout physical memory, eliminating core shuffling and complex core management - even if the user's program expands during execution.

- The concept of memory mapping creates the hardware basis for "virtual memory" operation - the ability to execute a program when all of its "pages" are not contained in physical memory. If the program attempts to access a page not located in physical memory, a "page fault" occurs. The Monitor then causes the page to be swapped into main memory (from disk) and allows the program to continue operation.

A full discussion of the implementation of memory management hardware for the TOPS-10 and TOPS-20 operating system is beyond the scope of this document. This information is included in the KL Technical Summary and the TOPS-20 Operating System Self-Pace Sales Training Document.

THE FRONT-END SUBSYSTEM

A key element in the operation and maintenance of the KL, under both TOPS-10 and TOPS-20, is the PDP-11-based front-end computer. This console/diagnostic/unit record minicomputer provides all console functions for the KL and its associated operating systems, allows remote and local maintenance of the system, and supports several card readers and line printers.

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

In addition to the DTE link, both the KL (via an RH20) and the PDP-11 (via an Rh11) are interfaced to individual ports of an RP06 Disk Drive. Figure 4-1 illustrates these interconnections.

In addition to the console and diagnostic functions, the DTE/PDP-11 allows unit record and asynchronous communications equipment support via the PDP-11-based front-end computer. These devices operate in an identical manner (from a user standpoint) to those interfaced to individual PDP-11s, and are explained in detail in the following Section on I/O.

All KL-based systems rely on the front end for at least two basic functions:

- Initiation of CPU operations from a dead stop. This process involves setting up KL status, loading microcode, configuring memory, and starting the Monitor bootstrap.

- Support of the console terminal, which allows the operator to control the system. This terminal directs the operating system and the tasks running under it.

These functions are implemented by a special operating system that runs in the PDP-11, with supporting code in both TOPS-10 and TOPS-20. The PDP-11-based operating system differs in various system models, but is built around Digital's popular and successful RSX (Real-Time System Exec). This software allows concurrent operation of multiple tasks.

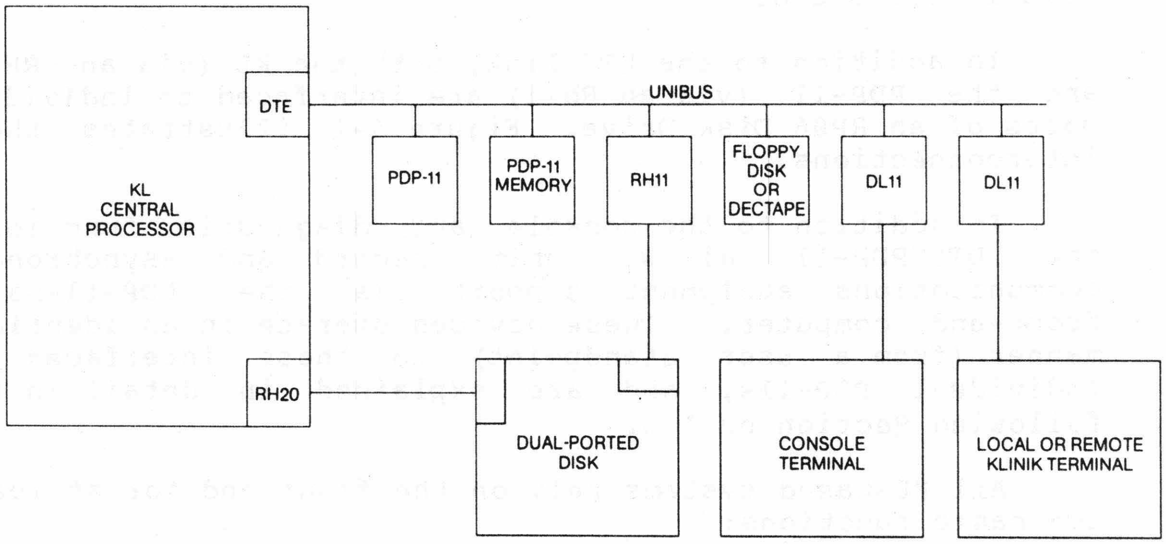


FIG. 4-1. RP06 DISK DRIVE AND CONSOLE FRONT-END INTERCONNECTIONS

Front-end operations require the connection of the dual-ported RPØ6 to both the KL and the PDP-11. The disk used is KL-formatted. There are both hardware and software interlocks to prevent the KL and PDP-11 from interfering with one another. The PDP-11 disk area is not available for user storage.

Depending on the system model, there may be additional devices associated with operation of the front-end subsystem. These include an RH11 disk controller and either a floppy disk or DECtape drive. The floppy disk or DECtape drive may function as an alternate bootstrap device, if the RPØ6 cannot be used or contains incorrect information. Users cannot operate these devices as normal peripheral devices.

The KLINIK capability of the KL is an attractive and powerful selling tool. The concept of "remote diagnosis" is fully supported by both hardware and software. Through special "diagnosis buses", the PDP-11 can diagnose a completely inoperative KL processor. In addition, special buses are provided so that diagnostics can be run during normal operations, without interfering with user operations. Spend a few minutes with your local Field Service manager to talk about KLINIK. Its success is well-published and frequently updated.

This page is for notes.

THE I/O SUBSYSTEM

The KL Input/Output subsystem includes standard interfaces and control logic that allow the connection of a wide range of peripheral and communications equipment to the KL. Exact I/O configuration depends upon the system model and operating system.

The peripherals supported use Digital's Corporate "BUS" architecture: the UNIBUS, the MASSBUS, and the traditional DECsystem-1Ø I/O BUS. Four bus systems are provided on the KL; however, not all are implemented in every system model and configuration. For example, TOPS-2Ø does not support the multiplexed I/O bus nor the DMA (for external memory). Figure 5-1 illustrates this bus architecture for the KL.

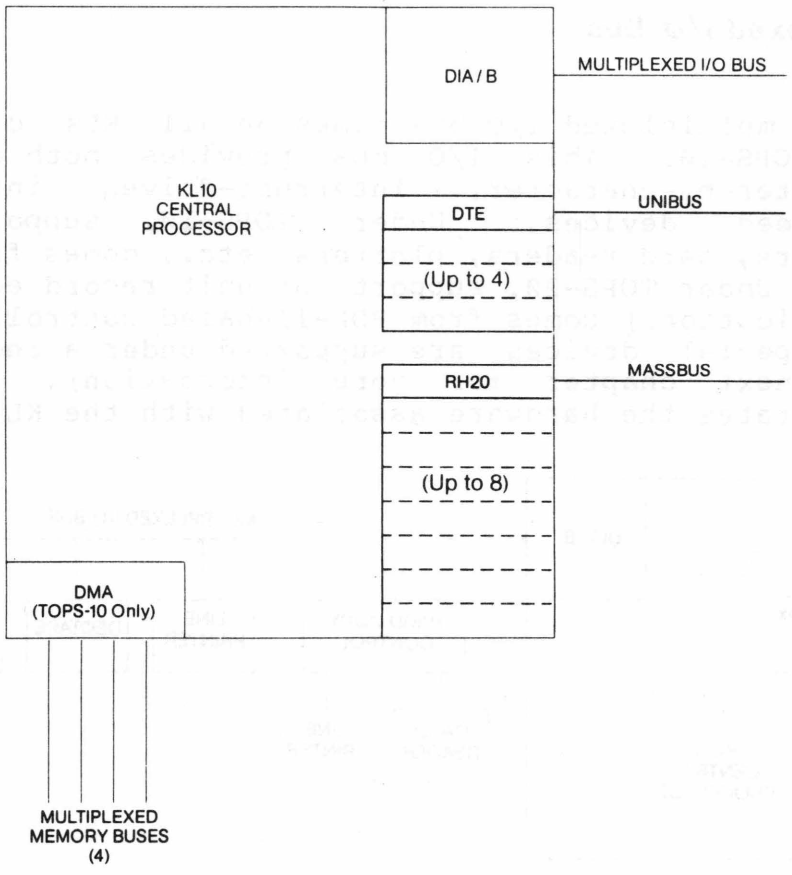


FIG. 5-1. KL10 BUS ARCHITECTURE

As illustrated in Figure 5-1, "outside world" devices may be attached to four external buses. These are:

- The Multiplexed I/O Bus - interface DIA/DIB
- The UNIBUS - interface DTE
- The MASSBUS - interface RH2Ø
- The Multiplexed Memory Bus - interface DMA

Each is discussed in some detail.

Multiplexed I/O Bus

A multiplexed I/O bus comes on all KLs configured to run TOPS-1Ø. This I/O bus provides both control and a character-by-character, interrupt-driven interface for low-speed devices. Under TOPS-1Ø, support for line printers, card readers, plotters, etc., comes from the I/O bus. Under TOPS-2Ø, support for unit record equipment (and communications) comes from PDP-11-based controllers, but a few special devices are supported under a recent release. (See next chapter for more information). Figure 5-2 illustrates the hardware associated with the KL I/O bus.

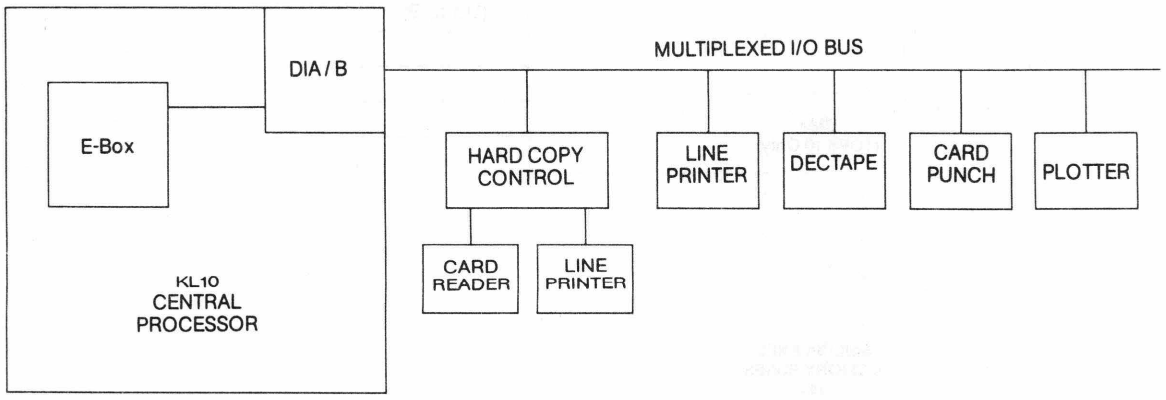


FIG. 5.2. MULTIPLEXED I/O BUS ARCHITECTURE

(Note: the DIA is configured in "tall", blue DECsystem-1Ø-style cabinets; the DIB is configured in Corporate cabinets, resembling the DECSYSTEM-2Ø and DECsystem-1Ø91). This I/O bus capability has an important benefit: the KL is a "drop-in" replacement for earlier KA and KI processors and their I/O bus architecture.

Memory Bus Subsystem

The KL processor supports an external memory bus system, which allows external memory to be connected to the processor. This external memory is supported under TOPS-1Ø only. The interface between the external memory and the M-Box is a four-word-wide DMA. This interface allows connection of up to 16 memory modules and up to 4,Ø96K words of external memory. Figure 5-3 illustrates memory configurations. The DMA allows memory modules of any speed and size to be connected to the KL. This DMA (coupled with the DIA I/O Bus) allows KL drop-in replacement for KAs and KIs, with the use of existing memory and peripherals.

The four external memory buses also support direct-to-memory channels and controllers once sold by Digital. These channels and controllers include interfaces to disks, tapes, and communications equipment. When such devices are already installed, and your customer desires to retain them for a KL upgrade, involve your Field Service manager early in the sale. This assures a smooth configuration and installation. Certain versions of TOPS-1Ø do not support older processors and peripherals; consult your local Software Support specialist for details.

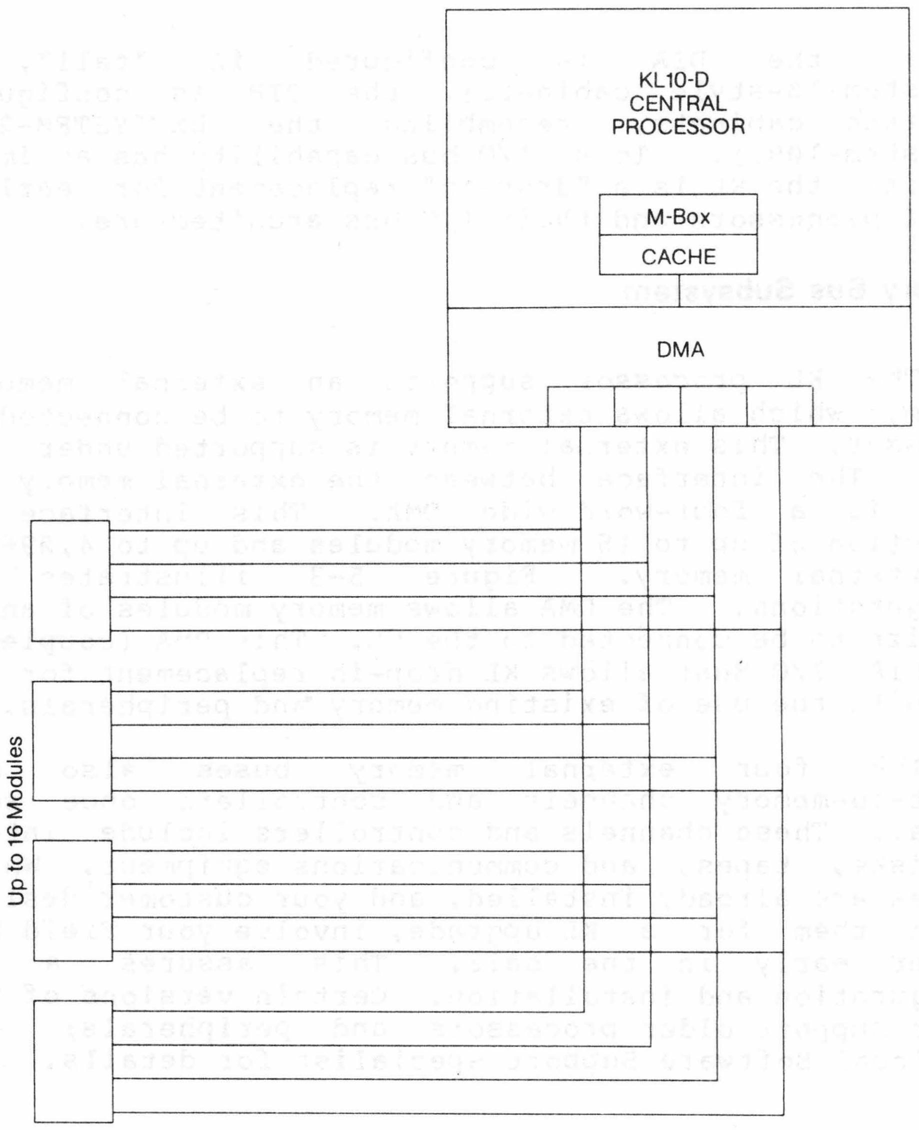


FIG. 5-3. KL 10-D MEMORY BUS STRUCTURE

MASSBUS

The KL processor interfaces to high-speed peripheral devices via Digital's popular and flexible MASSBUS. Each MASSBUS can interface up to eight peripheral devices/controllers. The MASSBUS architecture is provided by an RH2Ø integrated channel /controller interface. These RH2Øs plug directly into pre-wired slots in the CPU I/O section, and interface with the E-Box (for control) and one of eight data channels located in the M-Box (for data transfer).

Each RH2Ø channel also includes a 16-word "silo" buffer (first in-first out) to support high-speed, block transfer I/O. The buffer prevents "overruns", which result when the I/O device inputs data to the KL faster than the M-Box can dispose of (or store) it.

Up to eight RH2Øs may be installed on the KL. For each device, the channel provides the 15-word data buffer (or silo), a channel command-word register and a control command-word pointer. The channels transfer data by executing channel programs, set up in memory by the operating system. The memory set-up is accomplished using memory cycles, without any attention from the KL execution logic (E-Box), which allows overlapped I/O and computation.

Each RH2Ø terminates in a MASSBUS. Digital-supplied disks and tapes then connect directly to the MASSBUS, providing an integrated, high-speed subsystem for mass storage. Figure 5-4 reflects these connections.

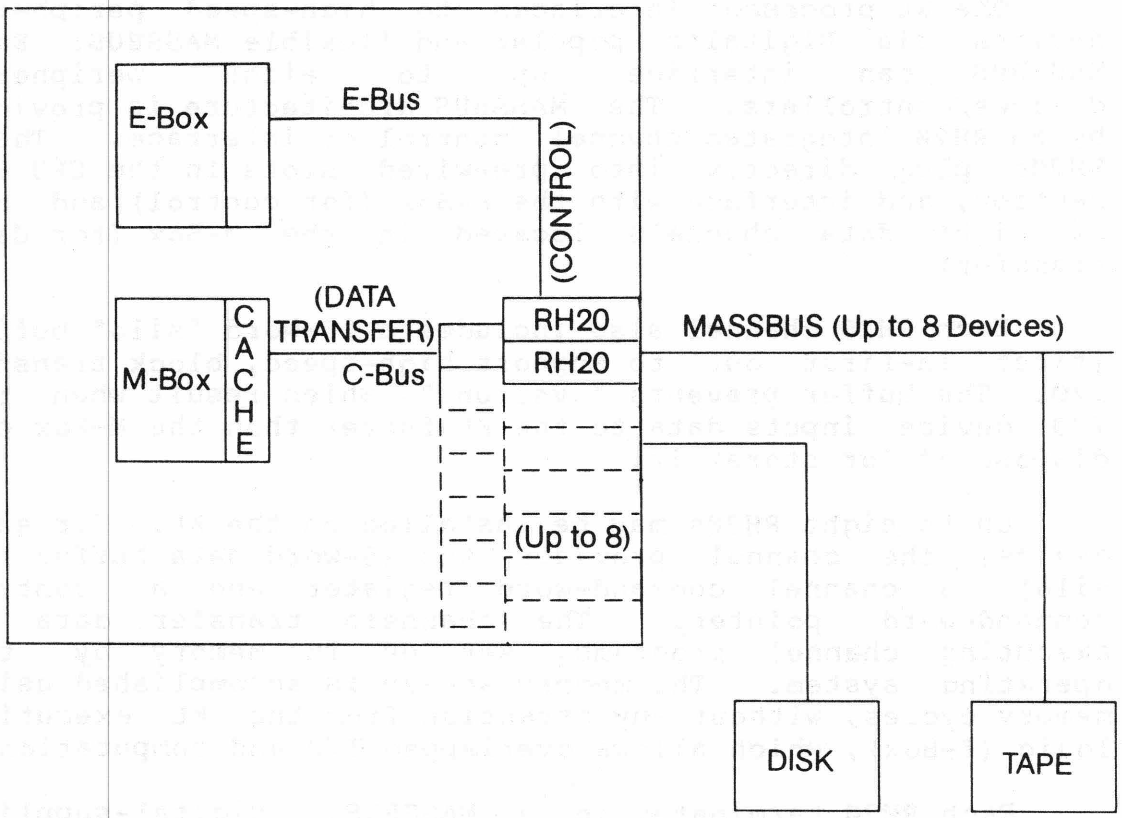


FIG. 5-4. MASSBUS INTERFACE

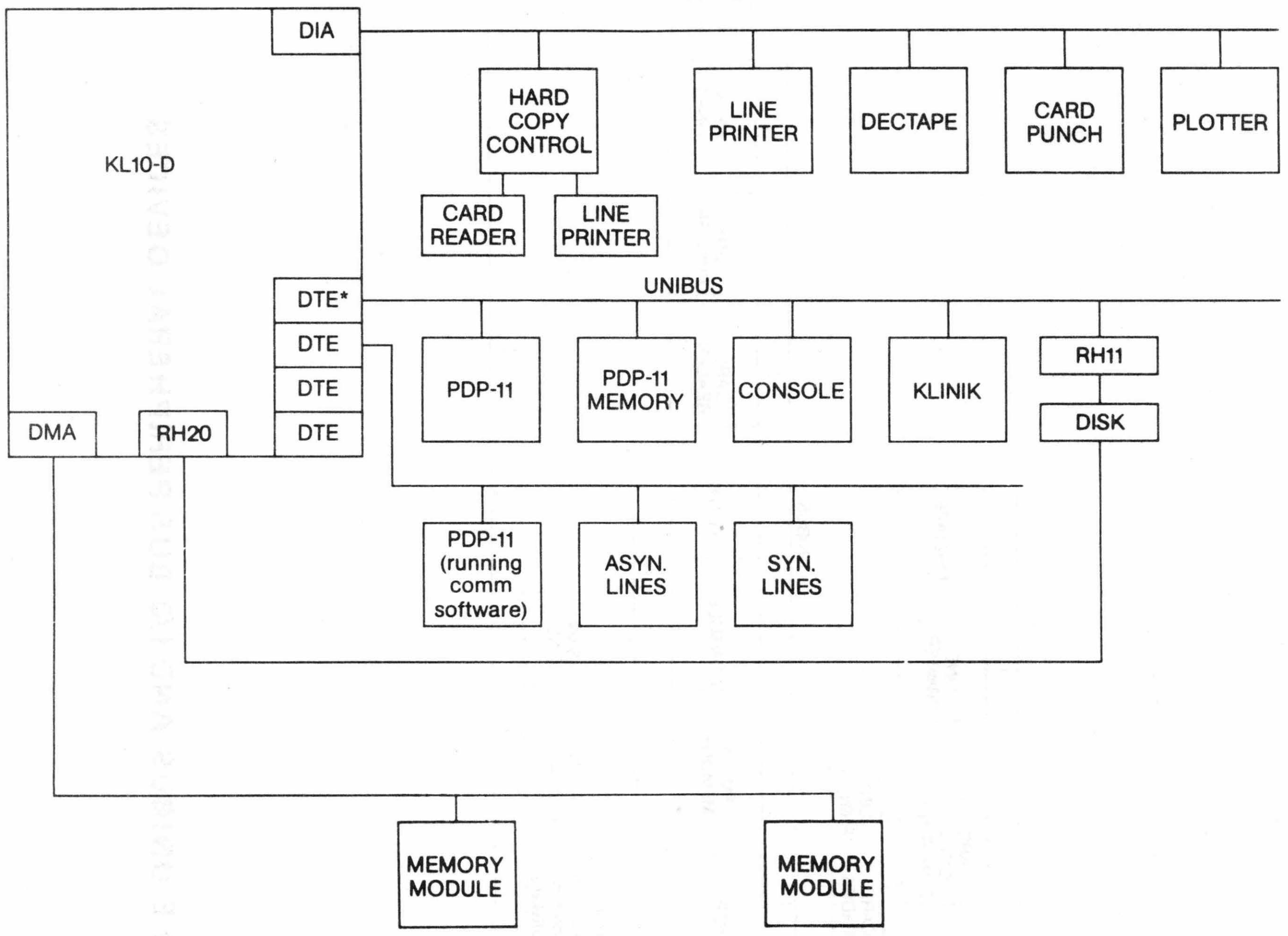
UNIBUS

Digital's popular and successful UNIBUS is the fourth path between external devices and the KL processor. As illustrated in Figures 5-5, 5-6 and 5-7, this interface utilizes the DTE (Digital Ten-to-Eleven) interface. The DTE is also used for the front-end subsystem, as explained in a previous section of this document. The front-end DTE is a privileged DTE; it allows special privileges connected with system start-up and diagnostic functions. The privileged DTE actually has access to internal buses on the KL. Only the restricted DTE, which is used to control peripheral devices and communications equipment, is discussed here.

Through the DTE, a PDP-11 can read or write a single KL word on its UNIBUS. In other words, it performs memory location, although the path is via the M-Box and cache on models equipped with cache memory. The PDP-11 may also operate in a byte-transfer mode, in which the DTE is responsible for transferring a string of data to or from KL memory. It is important to note that the PDP-11 itself is a UNIBUS device, and the DTE can access it. To protect system integrity, the operating system establishes "windows" for the areas within KL memory where word transfers may take place. This permits the KL to protect itself from a malfunctioning PDP-11.

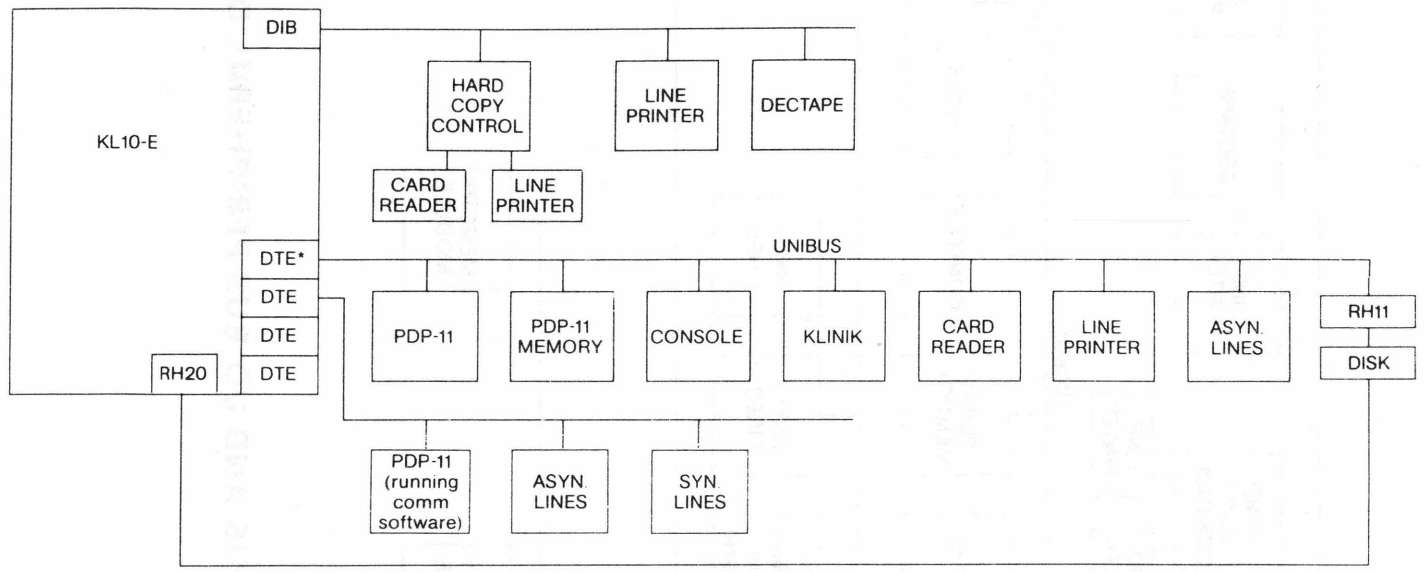
Once initiated by the operating system, a block of byte-transfer takes place without any further programmed intervention. On the PDP-11 side, the transfer is handled directly by the DTE and PDP-11 memory. On the KL side, the transfer is accomplished through microcode instructions. Thus, during transfers through DTEs, cycles are being stolen from the microcode. This results in a certain amount of CPU overhead.

Each DTE requires a PDP-11 processor for operation. Although service routines are integral to the operating system, it is the PDP-11 program that actually handles peripheral equipment and controls its operation. Under both TOPS-1Ø and TOPS-2Ø, card readers, line printers, and both synchronous and asynchronous communications equipment are handled by PDP-11s (communicating with the KL via the DTE). Hardware device interfaces are typically Corporate UNIBUS devices. Figures 5-5, 5-6, 5-7 illustrate a typical unit record/communications configuration for a KL running TOPS-2Ø, and TOPS-1Ø. Since marketing and packaging policies change with time, you should consult the appropriate Pricing Summary or Configuration Guide to determine which devices are supported on which models, for specific customer situations.



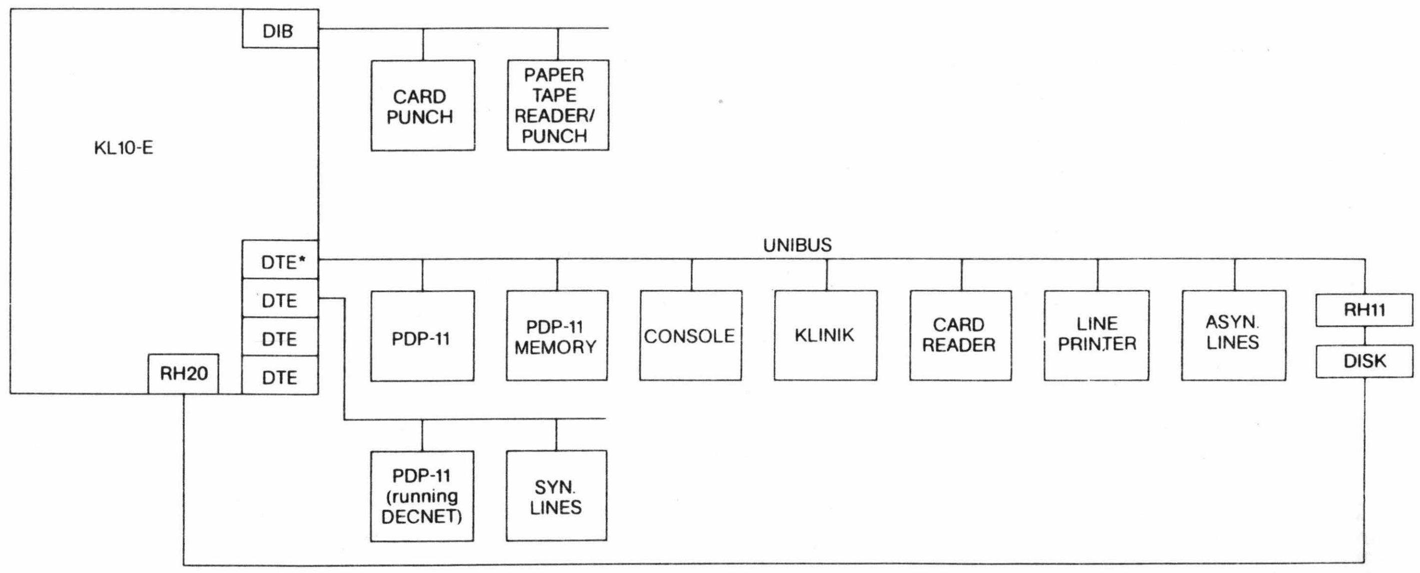
* Privileged DTE.

FIG. 5.5. KL10-D UNIBUS AND I/O BUS PERIPHERAL DEVICES



* Privileged DTE

FIG. 5.6. KL10-E UNIBUS AND I/O BUS PERIPHERAL DEVICES



* Privileged DTE.

FIG. 5.7. KL10-E UNIBUS AND I/0 BUS PERIPHERAL DEVICES

This page is for notes.

FOREWORD--A RICH HERITAGE OF SUCCESS

READ THE INTRODUCTORY SECTION OF THE TOPS-1Ø TECHNICAL SUMMARY. AFTER COMPLETING THIS READING (1 PAGE), READ THE PARAGRAPHS BELOW. THEN ANSWER THE SELF-CHECK QUESTIONS IN THIS STUDY GUIDE.

In the early 196Ø's, Digital began work on the hardware and software products needed to provide a "multi-user, timesharing" computing system. This work was influenced by Bolt, Beranek and Newman (a Cambridge consulting firm), who implemented a simple timesharing system on Digital's first computer system, the 18-bit PDP-1. This system used a high-speed drum that could swap a 4K word core image in one 34 millisecond revolution. The support hardware for such a timesharing environment was designed to support an operating system that would allow several users to run concurrent jobs on the machine--each apparently having sole control of the machine and its resources.

The initial goals, constraints and basic design concepts are outlined in Table 1. The primary goal was to provide a general-purpose timesharing system. The Monitor would allow the user to run in the mode most suited to his requirements, including interactive timesharing (from terminals), real-time (for laboratory data acquisition) and batch processing.

In this primitive timesharing environment, each user would receive adequate protection from other nonmalicious users. However, self-protection was not included, since the user could alter various user status information. Because common system resources were allocated upon demand and deadlocks could occur, the term "Gentlemen's Timesharing" was coined for the first PDP-6 Monitor.

The original PDP-6 Monitor was less than 6K words! The Monitor has increased at about 25 percent per year, with the KA1Ø at 3ØK words, the KI1Ø at 5ØK words, and the KL1Ø at 9ØK words. This increase provided expanded functionality (e.g., better files, batch, automatic spooling, etc.), larger system configuration size, more I/O options, increased number of jobs, easier system generation and increased reliability (e.g., checking, retries, file backup, etc.) Table 2 outlines the evolution of the TOPS-1Ø Monitor to its current functional state in several key areas.

Table 1. Initial Goals, Constraints and Basic Design Decisions

---

User/Language/Operating System

- Cheap cost per user via timesharing without inconvenience of batch processing
- Timeshared use via terminals with protection between users
- Independent user machines to execute from any location in physical memory
- Unrestricted use of devices, e.g., full-duplex use of terminals
- Support for wide range of compiled and interpreted languages
- No special batch mode; batch must appear like terminal via a command file
- Device-independent I/O; programs run on different configurations and I/O is shared among the user community
- Direct I/O for real-time users
- Primitive command language to avoid need for large internal state
- Minimum usable system <16K words
- Modular software corresponding to modular hardware configurations

Organizational/Marketplace

- Add to high end of DEC's computers
- Use minimal resources, while supporting DEC's minicomputer efforts

Table 2. Monitor Functions Evolution

Facility	PDP-6(1964)	KA1Ø(1967)	KI1Ø(1972)	KL1Ø(1975)
Protection	One segment per user	Two segments with shared program segment (required re-entrant programs,	Four modes for shared segments	Virtual machine with shared segments
Program swapping	Core shuffling	Core shuffling with swapping (via drum disk)	Paging used for core management	Demand paging (job need not be wholly resident to run)
Facilities allocator	Devices assigned to users upon request (deadlocks possible=> gentlemen's timesharing)	Spooling of line printer and card reader	Spooling of all devices	
Scheduler	Round-robin scheduler	Scheduler to favor interactive jobs using multiple queues	Fairness and swapping efficiency considerations	Parameters for scheduling set by system manager; priority job classes and "pie-slice" schedule
User files	User files on DECtape, cards and magnetic tape	Significant enhancement of file function; on-line, random access disk-based files	Improved file structure reliability, error recovery, protection and sharing; mountable structures	Disk head movement optimization

Table 2. Monitor Functions Evolution (continued)

Facility	PDP-6(1964)	KA1Ø(1967)	KI1Ø(1972)	KL1Ø(1975)
Command control program	Simple (to implement) requiring little state	Evolution to more powerful, easier to use command language	Common Command Language (CCL)	Extensions to CCL
Batch	No real batch	Remote and local single-stream batch	Multiprogramming batch	Improved multiprogramming batch
Terminal handling and communications	Asynchronous task-to-task communications (for interactive terminals) as Monitor module	Synchronous communications for remote job and concentrator stations; "birth" of networks with simple topologies; ARPA network	Synchronous communications in complex topologies; new protocol; IBM BISYNC for 278Ø emulation/termination	DECnet communications*
Multiprocessing		Dual processor support (master/slave)	High availability through bus switching hardware	Symmetric multiprocessing (1979)

*DECnet is DEC's computer network protocols and functions