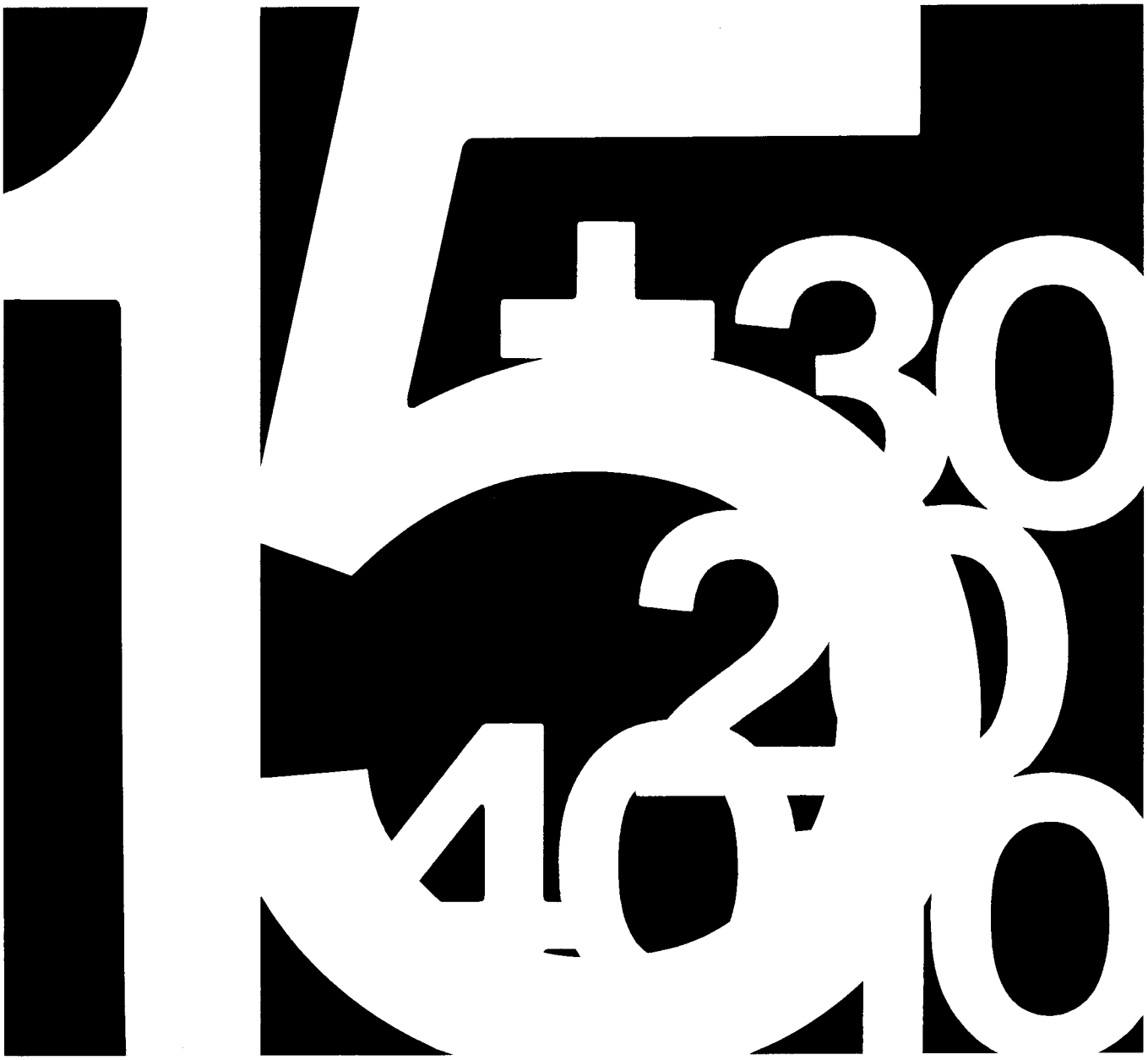
Digital Equipment Corporation
Maynard, Massachusetts

digital

Reference Manual

PDP-15 Systems



DEC-15-BRZA-D

PDP-15
SYSTEMS REFERENCE MANUAL

DIGITAL EQUIPMENT CORPORATION ▫ MAYNARD, MASSACHUSETTS

1st Printing August 1969

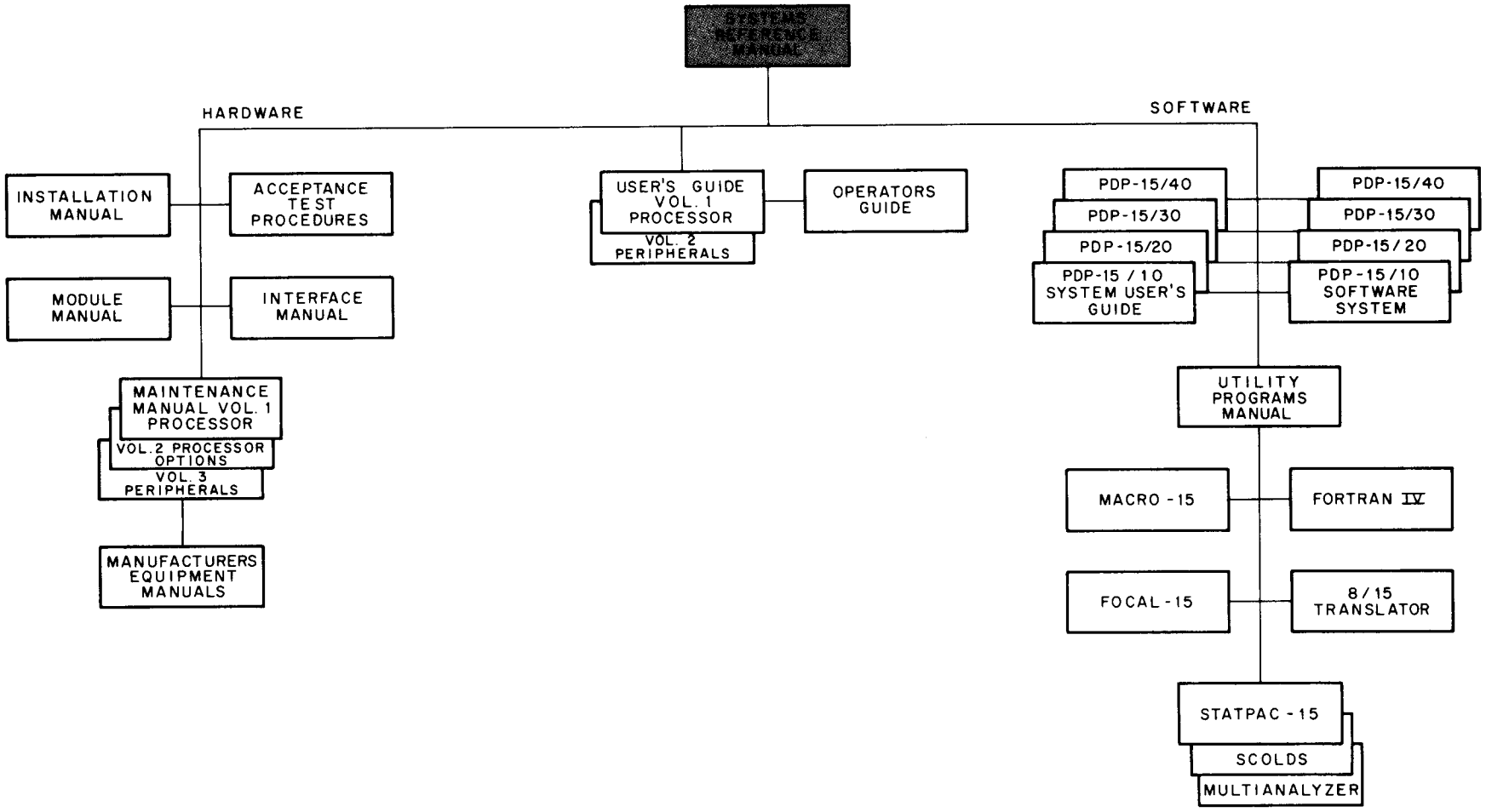
Instruction times, operating speeds and the like are included in this manual for reference only; they are not to be taken as specifications.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

TABLE OF CONTENTS (cont.)

PDP-15 FAMILY OF MANUALS



15-0040

SYSTEM REFERENCE MANUAL - Overview of PDP-15 hardware and software systems and options; instruction repertoire, expansion features and descriptions of system peripherals.

USERS GUIDE VOLUME 1, PROCESSOR - Principal guide to system hardware includes system and subsystem features, functional descriptions, machine-language programming considerations, instruction repertoire and system expansion data.

VOLUME 2 PERIPHERALS - Features functional descriptions and programming considerations for peripheral devices.

OPERATOR'S GUIDE - Procedural data, including operator maintenance, for using the operator's console and the peripheral devices associated with PDP-15 Systems.

PDP-15/10 SYSTEM USER'S GUIDE - COMPACT and BASIC I/O Monitor operating procedures.

PDP-15/20 SYSTEM USER'S GUIDE - Advanced monitor system operating procedures.

PDP-15/30 SYSTEM USER'S GUIDE - Background/Foreground monitor system operating procedures.

PDP-15/40 SYSTEM USER'S GUIDE - Disk-oriented background/foreground monitor system operating procedures.

PDP-15/10 SOFTWARE SYSTEM - COMPACT software system and BASIC I/O Monitor system descriptions.

PDP-15/20 ADVANCED Monitor Software System - ADVANCED Monitor System descriptions; programs include system monitor and language, utility and application types; operation, core organization and input/output operations within the monitor environment are discussed.

PDP-15/30 BACKGROUND / FOREGROUND Monitor Software System - Background/Foreground Monitor description including the associated language, utility and applications programs.

PDP-15/40 Disk-Oriented BACKGROUND/ FOREGROUND Monitor Software System - Background/Foreground Monitor in a disk-oriented environment is described; programs include language, utility, and application types.

MAINTENANCE MANUAL VOLUME 1, PROCESSOR - Block diagram and functional theory of operation of the processor logic. Preventive and corrective maintenance data.

VOLUME 2, PROCESSOR OPTIONS - Block diagram and functional theory of operation of the processor options. Preventive and corrective maintenance data.

VOLUME 3, PERIPHERALS (Set of Manuals) - Block diagram and functional theory of operation of the peripheral devices. Preventive and corrective maintenance data.

INSTALLATION MANUAL - Power specifications, environmental considerations, cabling and other information pertinent to installing PDP-15 Systems.

ACCEPTANCE TEST PROCEDURES - Step-by-step procedures designed to insure optimum PDP-15 Systems operation.

MODULE MANUAL - Characteristics, specifications, timing and functional descriptions of modules used in PDP-15 Systems.

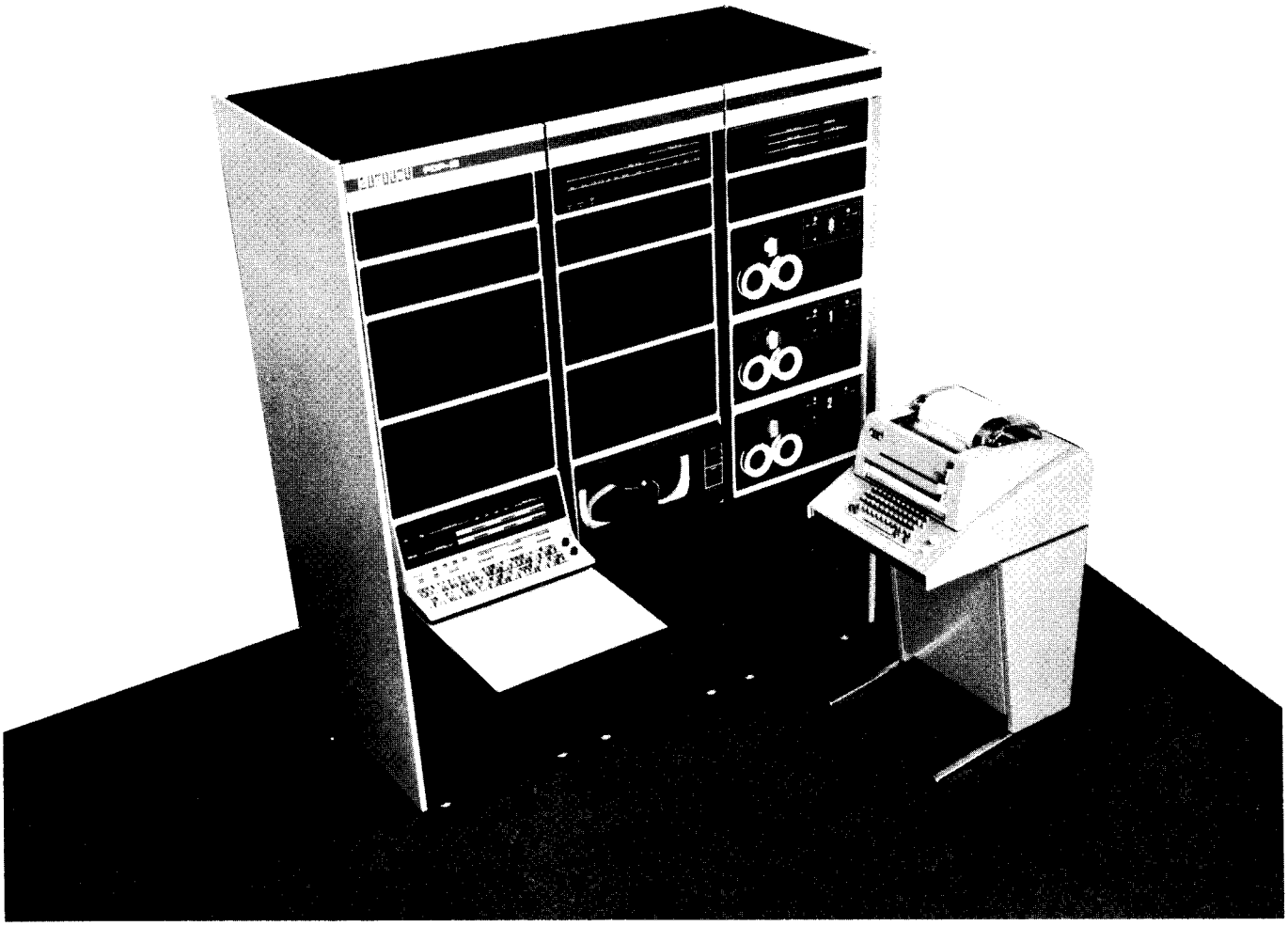
INTERFACE MANUAL - Information for interfacing devices to a PDP-15 System.

UTILITY PROGRAMS MANUAL - Utility programs common to PDP-15 Monitor systems.

MACRO-15 - MACRO assembly language for the PDP-15.

FORTRAN IV - PDP-15 version of the FORTRAN IV compiler language.

FOCAL-15 - An algebraic interactive compiler-level language developed by Digital Equipment Corporation.



PDP-15 System

FOREWORD

PDP-15 Systems offer comprehensive solutions to real-time data problems. They combine new design concepts with a wide array of traditional features that spring from Digital's years of leadership in the medium-scale scientific computer field. Both elements share the common purpose of simplifying the user's tasks in a demanding real-time environment.

Since certain types of data-handling tasks require specific hardware and software configurations, Digital has developed four standard PDP-15 Systems, ranging in power from the modestly priced basic PDP-15/10 to the real-time disk monitor environment of the PDP-15/40. At every level, the capabilities of the hardware are under the control of a monitor designed specifically for them.

The software systems were designed around the hardware with the user environment in mind. The principal design objectives were to provide (a) a system that is convenient for the user to implement and that affords the user access to the full power of the hardware, (b) a system that allows the user to easily integrate his applications programs and special peripheral device handlers without forcing him to become a systems programmer and (c) a system that can expand naturally with any additional hardware the user purchases. PDP-15 Systems software allows the user to move from a very basic machine to a sophisticated system environment without the cost and complication of reprogramming at each upward step.

The hardware systems were designed with several functional objectives in mind. Among these are the complete autonomy between central processor, input/output processor, and memory, so that processing and I/O operations can occur concurrently in overlapping cycles; TTL integrated-circuit construction for high reliability; fast internal speeds, including an 800-ns memory cycle time, to meet the demands of real-time data processing; core memory expansion to 131,072 words for future growth; and a sophisticated memory-protect system for multi-user integrity. Peripheral device handling and interfacing to other instruments are easily accomplished, and system growth potential is virtually unlimited with the modular structure of PDP-15 Monitor systems.

SYSTEM DESCRIPTIONS

PDP-15/10: Basic System

The PDP-15/10 is the first level PDP-15 System. The system's design provides limited budget users access to the power, speed, and 18-bit word length of PDP-15 hardware, in the expectation that the system can later be expanded to take full advantage of the advanced software capabilities inherent in the system's design.

Hardware includes 4,096 18-bit words of core memory and a Model ASR33 Teletype console teleprinter. The system has the rapid PDP-15 800-ns memory cycle time which provides 1.6-μs add capability. Facilities for later expansion are prewired into the system; additional memory and peripherals can be plugged in as they are required.

Software is governed by the COMPACT Programming System, a complete package including Assembler, Editor, Octal Debugging Technique, and mathematical and utility routines. All are designed to function in a 4096 word system. The software offers complete upward compatibility at the source level and field-proven reliability. Programs written for execution under COMPACT may also run, with little or no modification, within all PDP-15 system levels up through PDP-15/30 and PDP-15/40 BACKGROUND/FOREGROUND systems.

PDP-15/20: Advanced Monitor System

PDP-15/20 is an 8,192-word mass storage oriented system designed for research and engineering environments where real-time data acquisition and control tasks are combined with program development and testing.

Program development, debugging, and modification are all handled under monitor control, virtually ending intermediate operations. Unique real-time input/output routines can also be integrated into the system monitor to accelerate set-up and recovery.

Users are spared the task of writing system software to handle input/outputs to all standard system peripherals, since appropriate routines are supplied with the monitor. The net result is that even inexperienced computer users can get their applications programs "on the air" in a minimum of time.

PDP-15/20 hardware facilities include not only 8,192-words of core memory and high-speed paper-tape facilities but also a DECtape control unit and two tape transports for convenient mass-memory storage. DECtape is Digital's compact, inexpensive answer to the problems of program and data storage. A single pocket-sized reel of tape can accommodate up to 150,000 words of information. The extra-heavy duty KSR35 Teletype unit is included in the PDP-15/20 configuration to guarantee a high degree of reliability under the strain of continued heavy use. Also included is the extended arithmetic element described in Chapter 3. This unit facilitates high-speed multiplication, division, shifting, normalization, and register manipulation.

The 15/20 Advanced Monitor System provides not only mass-memory supervisory control but also complete device independence, so that programs need not be limited to the use of certain specified input/output devices. Simple I/O statements control data handling; and the selection of physical devices is determined at run time on the actual machine, not when the program is written. This system also allows for easy integration of real-time I/O level subroutines as new devices are added.

The Advanced Monitor permits two types of user interaction. These are (a) batch processing for routine production jobs and (b) keyboard interaction so that the user operates the system with simple commands typed at the keyboard.

Other Advanced Monitor features which utilize processor options, include a real-time clock control and a priority interrupt control.

All Advanced Monitor functions, and the variety of additional system software routines available,

have a single aim: to make the system as approachable as possible to users who want "hands-on" interaction, yet as automatic as possible in terms of taking care of routine elements in programming.

PDP-15/30 BACKGROUND/ FOREGROUND System

The PDP-15/30 System was designed to meet the demands of research, engineering, and industrial environments, where one or more real-time tasks typically require continuous responsiveness from the computer but do not use 100% of its capacity.

Under the control of the PDP-15/30 BACKGROUND/FOREGROUND Monitor, real-time tasks are handled in the computer foreground and have immediate call on the system's resources via interrupts. Background time (time left over between service calls for the real-time tasks) is available for program development and testing or other lower priority computation.

The PDP-15/30 Background/Foreground system contains 16,384 words of core memory and all the devices standard for the PDP-15/20. In addition, PDP-15/30 Systems are equipped with a memory protect system, a real-time clock, automatic priority interrupt, a third DECtape transport and a second on-line teletype for background use.

The Background portion of the PDP-15/30 System encompasses the Advanced Monitor functions and capabilities: Macro assembler (MACRO-15), FORTRAN IV, FOCAL-15, EDIT, DDT (Dynamic Debugging Technique), DUMP for off-line debugging, PIP-15 for interdevice file transfers, S-GEN (System Generator), linking and loading, and interactive file access. In addition, the BACKGROUND/FOREGROUND Monitor contains all the supervisory controls necessary for concurrent processing of background and foreground tasks.

In addition to the Advanced Monitor programs and routines, PDP-15/30 users (and all other PDP-15 system users) can draw on the considerable program library and applications knowledge of DECUS (the Digital Equipment Computer Users Society), the second largest and most active computer users' group in the world. DECUS members share in the exchange of programs and technical papers at regularly scheduled meetings throughout the year. Proceedings of society meetings are published and distributed to members under Digital sponsorship.

PDP-15/40 Disk-Oriented BACKGROUND/ FOREGROUND System

PDP-15/40 Disk-Based Background/Foreground System fulfills the demands of industrial and engineering environments where the need for a background/foreground mode of operation is compounded by the necessity for large random-access files.

The PDP-15/40 System with 24,576 words of core memory, high-speed paper-tape facilities, and DECtape storage, also incorporates a DECdisk control and two random access DECdisk files. The two disks, whose storage capacity of 524,288 18-bit words can be expanded to 2,097,152 words, permit high-speed overlays chaining and system and user loading.

Other hardware features of the PDP-15/40 include a memory protect system, background Teletype and a real-time clock.

The Disk-Oriented BACKGROUND/ FOREGROUND Monitor System handles all the functions of the 15/30 BACKGROUND/ FOREGROUND Monitor in an open/ended high-speed disk environment.

CHAPTER 1
GENERAL DESCRIPTION

INTRODUCTION

Digital Equipment Corporation's PDP-15 Systems are 18-bit fixed word-length, general purpose, binary digital computers. These systems are highly reliable, flexible, and complex tools which, together with their monitor systems and line of peripherals, can be used to help solve such diverse problems as data acquisition, process control instrumentation, scientific computation and man-machine communications.

This manual furnishes the reader with enough background information to familiarize himself with the PDP-15 System's present and potential capabilities.

PDP-15 HIGHLIGHTS

Complete System Autonomy

The basic PDP-15 System has an organization consisting of three autonomous subsystems - Central Processor, Memory, and I/O Processor - each with independent timing and control logic. Communication between these subsystems is accomplished through the use of an effective, asynchronous, request scheme. This subsystem autonomy facilitates wide-scale expansion of the system, increased throughput, high capacity, reliability and maintainability. With this approach, there is no concern about obsolescence; new subsystems incorporating the latest developments in computer technology can be added readily, without affecting either program compatibility or the operation of other subsystems.

Central Processor Autonomy - The full capability for controlling and executing the stored program is centered in the system's Central Processor. The Central Processor, by virtue of its control autonomy, coordinates its own operation with that of other subsystems, thus providing supervisory control over the PDP-15.

As the main unit in this integrated control, the Central Processor contains arithmetic and control logic hardware for a wide range of operations. These include high-speed, fixed-point arithmetic with a hardware multiply and divide option, extensive test and branch operations implemented with special hardware registers, high-speed input/output instructions, and other arithmetic and control operations.

The basic processor includes a number of major registers for processor-memory communications, a program counter, an accumulator, an index register and a limit register. Two 18-bit registers are used for memory buffer functions. This allows for processor overlap with memory cycle time, and effects faster instruction execution times.

Memory Autonomy - Independent read/write control and buffer logic in each memory bank establishes complete autonomy for the memory. This means that memories of different cycle times can be used together, and also provides for the expansion beyond 32,768 words.

Input/Output (I/O) Processor Autonomy - The I/O Processor has prime responsibility for controlling intercommunications between external system devices and the PDP-15 Memory or Central Processor. This third autonomous subsystem of the PDP-15 System contains eight data channels which may use either single-cycle block transfer devices or multi-cycle devices. The I/O Processor also contains an addressable I/O bus and provisions to add a real-time clock and an automatic priority interrupt system.

The I/O Processor operates in parallel with the system's Central Processor and grants external devices access to PDP-15 Memory through the single- or multi-cycle data channels.

Real-Time Capabilities

The unique combination of a powerful processor and a very fast autonomous I/O section with such real-time features as an 8-level, 32 channel, automatic priority interrupt system coupled with a range of monitor systems, gives the user exceptional real-time capabilities on his PDP-15.

System Reliability

To ensure system reliability, options such as power fail detection, memory relocation, memory protection and memory parity are all available on the PDP-15 System.

Hardware Reliability, Maintainability and Low Cost

A combination of such engineering considerations as worst-case design criteria for propagation delays, over-voltage and over-current protection, point-of-load regulator cards, minimal cable lengths, and extensive control panel (where twenty-four 18-bit groups of signals can be displayed including all major registers and buses), proven TTL logic, single time-state stepping and automatic checkout procedures provide the PDP-15 with two key characteristics. It is an extremely reliable and easily maintainable digital computer.

SUMMARY OF PDP-15 CHARACTERISTICS

Description - Programmed Data Processor - fixed 18-bit word length, parallel mode, autonomous operation, TTL circuitry.

Instruction List - Five basic instruction groups:

Memory Reference
Operate
Input/Output Transfer
Register Control/Transfer
Extended Arithmetic Element

Core Memory - 800-ns magnetic core memory expandable to 32,768 words; expansion with 4096 word pages or 8192 word memory banks; independent bank read/write control; provision for expansion beyond 32,768 words to 131,072 words built in.

Fixed Point Arithmetic (1's and 2's Complement) - 18-bit ADD in 1.6 μs 18-bit multiply* in 7.0 μs, 18-bit divide* in 7.25 μs.

Addressing - Single level indirect addressing; direct addressing to 4,096 words, indirect addressing to 32,768 words, indexed to 131,072 words. In bank mode, direct addressing to 8,192 words and indirect to 32,768 words.

*With extended arithmetic element.

Indexing - 1 index register, 8 auto-increment locations.

Input/Output - Up to 8 bidirectional channels communicating directly between the I/O bus and memory; addressable I/O bus normally services up to 42 device controllers.

Interrupt System - Basic machine has one program interrupt line. As an option there is an 8-level (4 hardware and 4 software) 32-channel Automatic Priority Interrupt that is expandable to eight devices per hardware level; priority can be changed dynamically under program control.

System Console - Includes facility to display twenty-four 18-bit groups of signals including all major registers and buses of the processor and I/O section. These displays provide both programming and maintenance information.

Non-Existent Memory Trap - Accessing non-existent memory is sensed by a timer performing a watch-dog function. The user is informed of such errors through an API (level 0) or PI interrupt.

Real-Time Clock - Core location 00007 can be incremented by a user determined frequency. The real-time clock interrupts control the CPU when location 00007 overflows.

Peripheral Equipment - Includes paper-tape station, card readers, line printers, magnetic tape controllers, disk systems, displays, processors, A/D and D/A equipment and communications equipment.

Options - Extended Arithmetic Element, Power Fail, Memory Protection, Memory Parity, Memory Relocation, Real-Time Clock, Automatic Priority Interrupt system.

Operating Environmental Specifications

Equipment	Temperature	Relative Humidity
Processor	0° to 50°C	10% - 95%
Typical System*	10° to 35°C	25% - 75%

*Includes paper tape station, card reader, magnetic tape, line-printer, etc.

Power Requirements

Voltage (AC) (± 15%)	Frequency (Hz) (±2 Hz)	Average Current (A)
100	50	33
115	50	30
200	50	18
215	50	17
230	50	16
120	60	29
240	60	16

NOTE

Maximum average current includes the 7.5 A that two Type ASR or KSR Teletypes draw through the power control. The above figures represent average (maximum) line current of systems with memory, two Teletypes and maximum internal options connected.

CHAPTER 2
PDP-15 SYSTEM ORGANIZATION

Three autonomous subsystems - Central Processor, Memory, and I/O Processor - operating together under console control define the PDP-15 System. Coupled to the PDP-15 I/O Processor and serviced under the jurisdiction of the Monitor systems, an extensive line of peripherals including mass storage, displays, data communications and data acquisition equipment combine to form the PDP-15 System. Figure 2-1 illustrates the relationship of each computer subsystem and the peripherals to the entire complex.

CENTRAL PROCESSOR

The Central Processor functions as the heart of the computer by carrying on bidirectional communication with both Memory and the I/O Processor. Provided with the capability of performing all required arithmetic and logical operations, the Central Processor plays the major role in the control and execution of the stored program. It accomplishes this with an extensive complement of registers, control lines and logic gates.

MEMORY

The Memory, second of the three autonomous subsystems, is the primary storage area for computer instructions and system data. Memory is organized into pages which are paired into memory banks. Each page has 4096 18-bit binary words of high-speed random-access magnetic core storage. Each bank is an asynchronous unit of 8192 words; expansion capability to 32,768 words (four banks) is provided for each PDP-15 System. Any word in Memory can be addressed by either the Central Processor or the I/O Processor. The CPU has provisions to address up to 131,768 words of core memory. The autonomy of the memory system allows mixing banks with different cycle times.

I/O PROCESSOR

The third autonomous subsystem satisfies the peripheral data transfer needs. A diverse line of system peripherals available to the PDP-15 require this processor to interface three modes of input/ output:

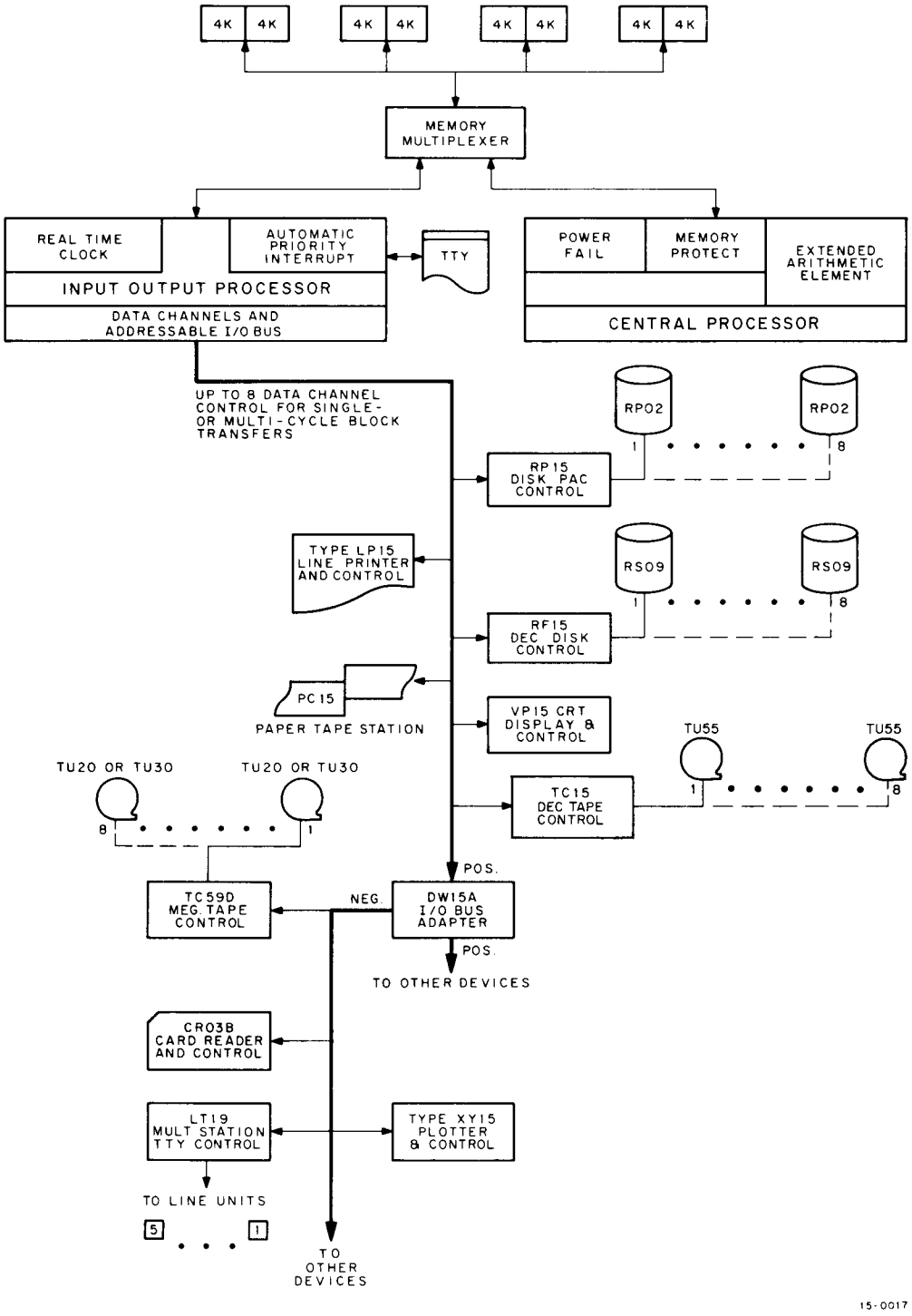


Figure 2-1. System Organization

Single cycle block data transfer; blocks of data transfer at rates up to one million words per second.

Multicycle block data transfer; blocks of data transfer at rates up to 250,000 words per second for input and 181,000 words per second for output.

Program controlled data transfers; single word transfers to/from the accumulator in the Central Processor

The I/O Processor provides timing, control and data lines for information transfers between memory or the Central Processor and the peripheral devices, it also includes provision for such options as the Automatic Priority Interrupt System and the Real-Time Clock.

SYSTEM PERIPHERALS

The PDP-15 System Peripherals range from simple input/output Teletypes to sophisticated interactive display processors. These peripherals communicate with the PDP-15 I/O Processor via one 72-wire bidirectional cable called the common I/O bus.

Figure 2-1 depicts the peripherals available with the PDP-15 Systems.

CHAPTER 3
CENTRAL PROCESSOR ORGANIZATION

SUMMARY OF CHARACTERISTICS

Description - 18-bit parallel operation, autonomous operation, fixed-point signed and unsigned arithmetic (1's and 2's complement)

Instruction Types -

Memory Reference
Operates
Register Transfer and Control
Extended Arithmetic Element
Input/Output Transfer

Indexing - 1 index register, 1 limit register, 8 auto-increment locations

Timing -

Typical Instructions	Execute Time
18-bit ADD	1.6 μs
18-bit Multiply*	7.0 μs
18-bit Divide*	7.25 μs
36-bit Shift*	7.25 μs
36-bit Normalize*	7.25 μs

*With EAE

CENTRAL PROCESSOR DESCRIPTION

The Central Processor (CPU) is the nerve center for control and execution of stored programs. By coordinating its own operation with that of other subsystems, it provides supervisory control over the PDP-15 System.

The Central Processor contains arithmetic and control logic hardware for a wide range of operations. These include high-speed fixed-point arithmetic with a hardware multiply and divide option, extensive test and branch operations implemented with special hardware registers, high-speed input/output instructions and other arithmetic and control operations.

The PDP-15 Central Processor contains several major registers for processor-memory communications, a program counter, an instruction register, an accumulator, an index register, and a limit register.

The CPU performs calculations and data processing in a parallel binary mode through step-by-step execution of individual instructions. Both the instructions and the data on which the

instructions work are stored in the core memory of the PDP-15. The arithmetic and logical operations necessary for the execution of all instructions are performed by the arithmetic unit operating in conjunction with central processor registers. Figure 3-1 shows a simplified block diagram of the Central Processor.

Arithmetic Unit (AU)

The PDP-15 arithmetic unit handles all Boolean functions and contains an 18-bit, 100-ns adder. The arithmetic unit acts as the transfer path for inter-register transfers and shift operations.

Instruction Register (IR)

Accepts the six most-significant bits of each instruction word fetched from memory. Of these bits, the four most-significant constitute the operation code, the fifth signals when the fetched instruction indicates indirect addressing, and the sixth indicates indexing.

Accumulator (AC)

This 18-bit register retains the result of arithmetic/logical operations for the interim between instructions.

For all program-controlled input-output transfers, information is transferred between core memory and an external device through the AC. The AC can be cleared and complemented. Its contents can be rotated right or left with the Link (see below). The contents of the memory, buffered through the memory input register, can be added to the contents of the AC with the result left in the AC. The contents of both registers can be combined by the logical operations AND and exclusive OR, the result remaining in the AC. The inclusive OR can be performed between the AC and the accumulator switches on the operator console (through the data switch register) and the result left in the AC.

Data Switch Register

The data switch register receives an 18-bit word through the console bus from data switches on the console. This register buffers the information so that the console may either be attached to the processor or operated remotely through cabling.

Link (L)

This 1-bit register is used to extend the arithmetic capability of the accumulator. In 1's complement arithmetic, the Link is an overflow indicator; in 2's complement arithmetic, it logically extends the accumulator to 19 bits and functions as a carry register. The program can check overflow into the Link to simplify and speed up single- and multi-precision arithmetic routines. The Link can be cleared and complemented and its state sensed independent of the accumulator. It is included with the accumulator in rotate operations and in logical shifts.

Program Counter (PC)

The program counter determines the program sequence (the order in which instructions are performed). This 18-bit register contains the address of the memory cell from which the next instruction is to be taken. The least-significant 15 bits are used for addressing 32,768 words of core memory. The remaining 3 bits provide the capability to address memory systems greater than 32,768 words.

Operand Address Register (OA)

The operand address register contains the effective address of the location where data is currently being fetched.

Memory Input and Output Buffer Register (MI) and MO)

Information is read from a memory cell into the memory input register and is interpreted as either an instruction or a data word. Informa-

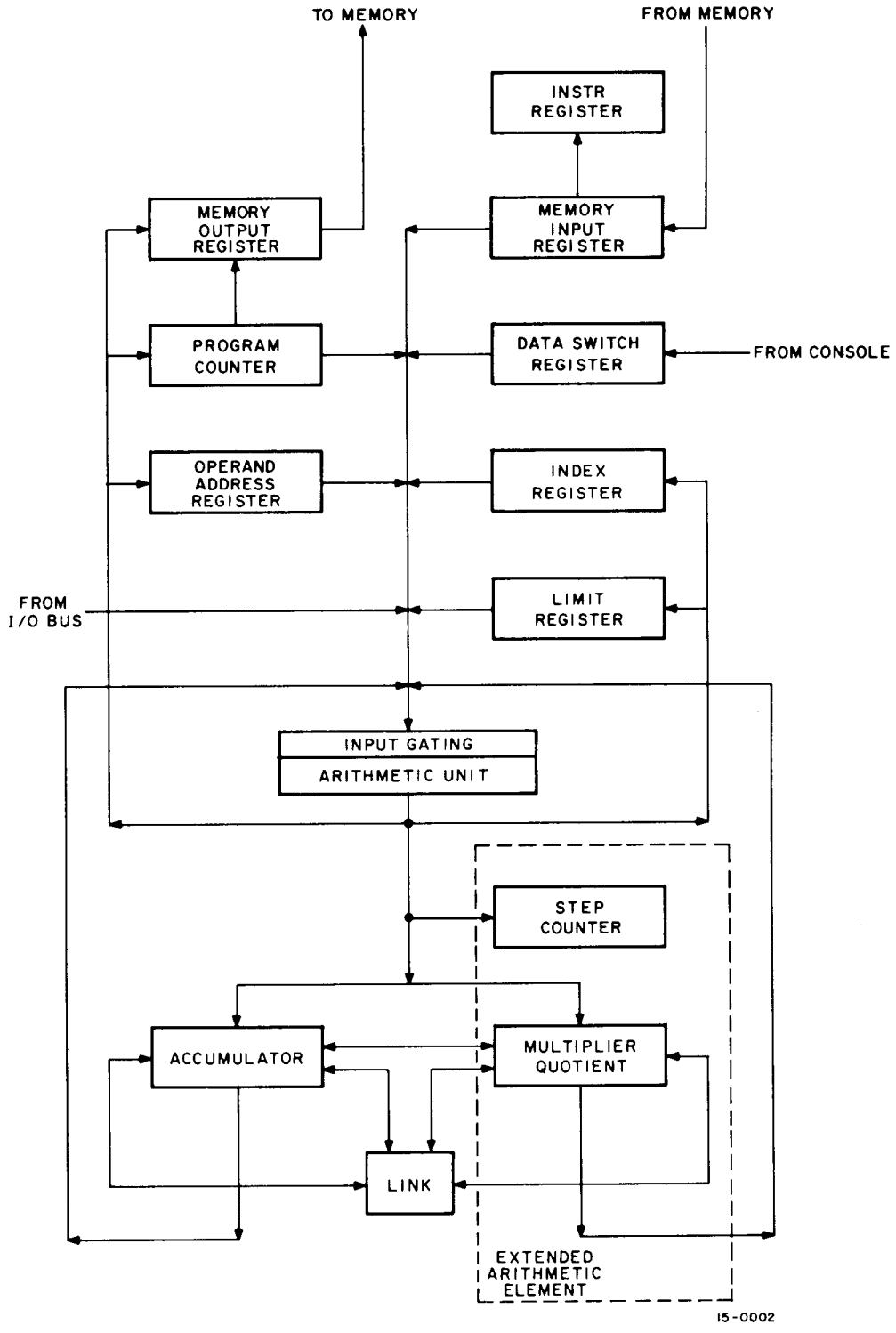


Figure 3-1. Central Processor, Simplified Block Diagram

tion is read from the Central Processor into memory through the memory output register and is interpreted as either an address or a data word. The use of two 18-bit registers for memory buffer functions allows processor overlap with memory cycle time to decrease execution time and to allow autonomous operation of the CPU and memory.

Index Register (XR)

This 18-bit register is used to perform indexing operations with no increase in instruction time. An indexed operation adds the contents of the index register to the address field of the instruction operand producing an effective address for the data fetch cycle. Index value can be positive or negative in 2's complement form (± 131,072).

Limit Register (LR)

The limit register enables a program to detect loop completion. The base address of a data array is loaded into the index register and the ending address is loaded into the limit register. Within an indexing loop, add to index and skip (AXS) instruction adds a signed value ($-256_{10} \leqslant Y \leqslant +255_{10}$) to the index register and compares the sum in the index to the contents of the limit register. If the contents of the index register are equal to or greater than those of the limit register, the next instruction is skipped. The limit register also provides a means for magnitude comparison of values between the accumulator and the limit register, through the use of a similar instruction, AAS (add to the accumulator and skip).

PDP-15 Control Console

The PDP-15's control console contains the keys, switches, and lights required for operator initiation, control, and monitoring of the system. Up to twenty-four 18-bit registers can be displayed to provide the user with visual indication of all registers and buses.

The console can be ordered in any of three different forms: a flush-mounted console which can be covered by cabinet doors for applications where "hands-off" security is paramount; a tilted console with table; or a remote console attached to the CPU by a single cable. The console can be remoted up to 100 ft.

Some of the features of the console are:

A READ-IN switch to initiate the reading of paper tapes; REGISTER indicators and REGISTER DISPLAY switches to allow continual monitoring of key points in the system such as the accumulator, index register, limit register, multiplier-quotient register, program counter, memory address, interrupt status, input/output bus, input/output address, and I/O status.

DATA switches to establish an 18-bit data or instruction word to be read into memory by DEPOSIT switch or to be entered into the accumulator by a program instruction.

EXAMINE switch to allow the manual examination of the contents of any memory location.

PROCESSOR EXPANSION

The following additional expansions extend processing capabilities of PDP-15.

Extended Arithmetic Element (EAE)

The Extended Arithmetic Element (standard on PDP-15/20/30/40 Systems) facilitates high-speed arithmetic operations and register manipulations. Installation of the EAE adds an 18-bit multiplier-quotient register (MQ) to the system as well as a 6-bit step counter register (SC). The multiplier-quotient register and accumulator perform as a 36-bit register during shifting, normalizing, and multiplication operations. The contents of the multiplier-quotient register are displayed by the REGISTER indicators on the

operator's console when the REGISTER DISPLAY control is in the MQ position. The option and the basic computer cycle operate asynchronously, permitting computations to be performed in the least possible time. Moreover, EAE instructions are microcoded so that several operations can be performed by one instruction to simplify arithmetic programming and reduce execution time. Worst case multiplication time is 7 μs; division time is 7.25 μs. The EAE is optionally available for the PDP-15/10.

Memory Protection

The memory protection feature, standard on PDP-15/30 and 15/40 Systems, establishes a background/foreground environment for PDP-15 processing activity by specifying the boundary between protected (lower) and unprotected (upper) regions of system core memory. Allocation of memory locations (in increments of 256 words) to the protected region is dynamic and program-controlled under the Background-Foreground Monitor. The protect feature increases all memory cycle times by 100 ns and write cycles in user mode by an additional 100 ns.

The protection option also provides a user/monitor mode of operation. When in user mode, attempted execution of any privileged instructions results in a trap to the monitor and a corresponding error message. These illegal instructions include input/output transfers and control, halts, chained executes, any references to the memory protect option itself, or protected memory. In monitor mode, all instructions are executable.

Power Failure Protection

The basic PDP-15 is not affected by power interruptions of less than 10-ms duration. Active registers in the processor may lose their contents when interrupts of longer duration occur, but memory is not disturbed. The Power Failure Protection option, available for all PDP-15 Systems, provides for saving the active register contents in the event of longer power interrupts and the automatic restart of the system when power is restored. When the line power failure occurs, the system must be operating with the program interrupt facility or the automatic priority interrupt system enabled in order to sense the Power Failure Protection's initiation of a program interrupt in time to save the register contents.

CHAPTER 4
MEMORY ORGANIZATION

SUMMARY OF CHARACTERISTICS

Speed	Cycle time - 800 ns Access time - 400 ns
Stack	Organization - 3-wire, 3D Core type - extended temperature 18 mil Drive Scheme - dc
Environment	Temperature - 0 to 50°C ambient
Special Features	Single Bus Type - Multiuser, bidirectional Parity (optional) Bank Selection
Cycle Types	Read - Restore Clear - Write Read - Pause - Write

CORE MEMORY

The magnetic core memory is the primary storage facility of the PDP-15. It provides rapid, random-access data instruction storage for both the Central Processor and the I/O Processor. The basic PDP-15/10 Memory contains 4096 18-bit word locations. The content of each location is available for processing in 400 ns. A parity bit can be added as an option to each word for parity checking during transfer of information into or out of core memory. If the parity option is incorporated into a PDP-15 System, all memory banks must contain that option and memory cycle time becomes 1.0 μs. The basic subsystem of Memory is the bank, which is organized into pages; each bank has two pages of 4096 words each for a total of 8192 words of 3D 3-wire cores. Further, every bank contains a data buffer, an address buffer and all the necessary read/write and control circuitry to make it an autonomous unit operating on a request/grant basis with either the Central or I/O Processor. Figure 4-1 illustrates the organization of a memory bank.

Memory Data Transfer

The PDP-15 Memory communicates directly with the Central Processor and the Input/Output (I/O) Processor through the memory bus. Data and instruction words of each bank are read from and written into individual memory cells through a buffered register referred to as the memory data buffer. (See the Memory Block Diagram, Figure 4-1.)

Words in a memory bank are selected according to the address in the memory address buffer. The 13-bit capacity of the memory address buffer allows 2^{13} or 8192 words to be referenced in each bank.

The memory address buffer receives the memory cell address from the Central Processor or I/O Processor. The address provides the coordinates for locating a word in a memory bank.

Decoding of the memory address to select a particular word location containing 18 bits is performed by the memory selection logic. Bit 5 of the memory cell address selects which page the cell is in, and the remaining bits select the X and Y coordinates of the cell.

Bits 1 to 4 of the memory bus select lines are used to select which bank of Memory the word is in. Up to 4 banks can normally be added to the PDP-15, but a special provision to expand memory up to 16 banks can be accommodated by the 18-bit address registers in the CPU.

Memory Cycles

Words are read from and written into Memory by a fixed sequence of events called a *memory cycle.* The memory cycle consists of a read half-cycle and a write half-cycle. Each type of half cycle requires 400 ns. Thus, the effective cycle time is 800 ns. For most applications, however, the two processors initiate a memory request and wait until the end of the read half-cycle only. At this time, the desired data is available for reading or has been accepted at the memory data buffer for writing, and the Central Processor or I/O Processor may proceed to the next step in its logical sequence of operations and perform useful functions while the write half-cycle is taking place. Thus, main memory access operations normally cause the two processors to wait only 400 ns. Delays caused by simultaneous requests by the two processors are discussed later in this section.

Read Half-Cycle - The read half-cycle copies the contents of the memory cell specified by the contents of the memory address buffer into the memory data buffer. If the parity option is present, a parity check bit is copied into the memory data buffer at the same time.

Write Half-Cycle - The write half-cycle copies the contents of the memory data buffer into the addressed memory cell. This half-cycle always occurs in conjunction with and following a read half cycle, although there may be a "pause" between them, during which the I/O Processor can manipulate the data in its Add-to-Memory mode (see I/O Data Processor Add-to-Memory Description, Chapter 5).

Parity

The parity option provides core planes that have 19 bits for each word and parity checking/generating control logic. When the parity option is present, the accuracy of transfers to and from Memory is verified by means of parity checking. A parity bit is added to each word stored in Memory such that the total number of 1 bits in the word, including the parity bit, is *odd.* For example, if the 18-bit word to be stored in Memory contains an even number of 1s, the parity bit is automatically made a 1, and is stored with the word. When the word is later read from Memory, the computed parity bit is calculated on the basis of the content of the 18-bit word. The two parity bits are then compared, if they do not agree, the memory parity error alarm is turned on, causing a program interrupt or automatic priority interrupt request, or simply a Halt.

All 18 bits and the accompanying parity-check bit (when present) are transferred in *parallel* (simultaneously) between the core array and the

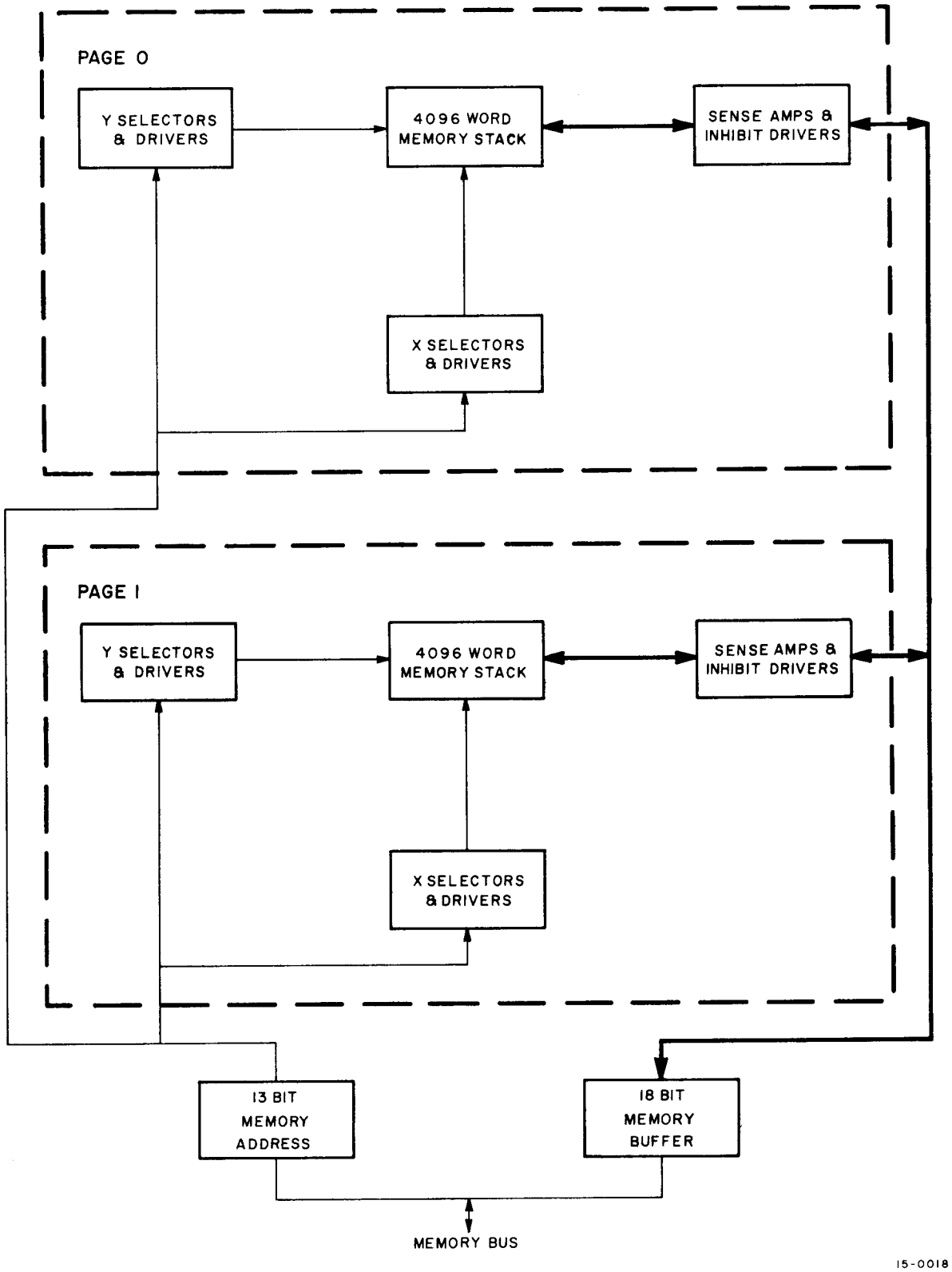


Figure 4-1. PDP-15 Memory Bank

memory data buffer, as shown in Figure 4-1. The memory data buffer is connected to the memory bus and hence to the rest of the PDP-15 System. This is also an 18-bit parallel transfer.

Memory Modularity

The PDP-15/10 System contains one page of 4096 memory words. Additional modules (pages) may be added to the system. The basic system can accommodate up to 32,768 core memory words (eight 4K pages) in the basic 19-in. rack mount cabinet. Expansion beyond 32,768 words requires the addition of another cabinet to the system configuration. Memory communicates with the central processor and the I/O processor on the bidirectional, interlocked party-line memory bus (See Figure 4-2).

Memory Multiplexer

The memory multiplexer allows both the Central and I/O Processors to share core memory. In the event that both request a memory cycle simultaneously, the I/O Processor will be serviced first and the Central Processor must wait. However, if only one processor is using Memory, then both can process at the same time. For example, the Central Processor may be executing an EAE instruction while the I/O Processor transfers data out of Memory to a DECdisk.

Memory Relocation

Memory relocation is optional on all PDP-15 Systems. This feature provides a relocation register and an upper boundary register to permit hardware relocation of both system and user programs. It allows the relocated program to execute only within its specified boundary; thereby providing protection for all other programs.

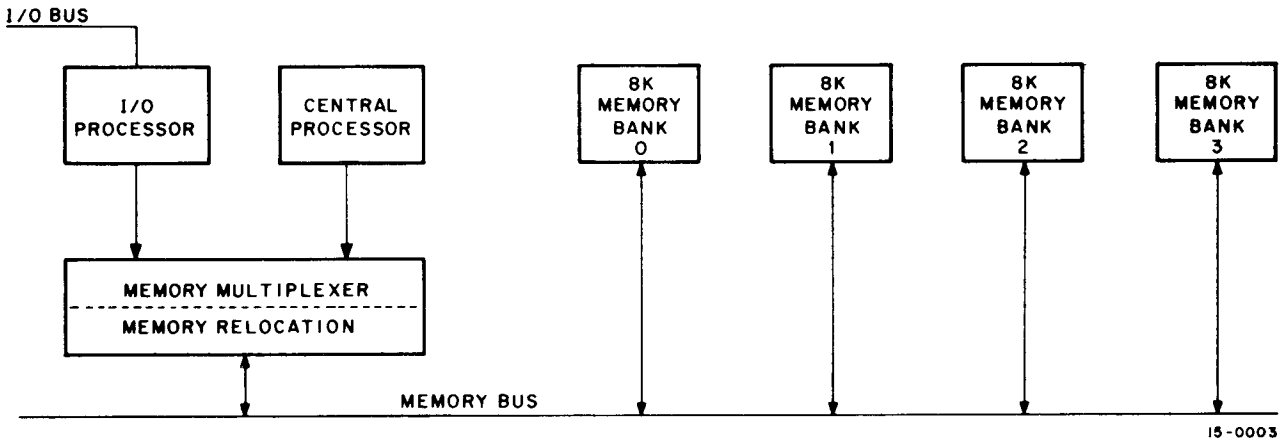


Figure 4-2. PDP-15 Memory Bus

CHAPTER 5
ORGANIZATION OF THE INPUT-OUTPUT PROCESSOR

SUMMARY OF CHARACTERISTICS

The I/O Processor contains two subunits, the data channel controller and the addressable I/O bus.

Data Channel Controller

Data Transfer Modes - Single and multicycle block transfer, memory-increment, add-to-memory

Data Channels - Eight standard

Options - Real-Time Clock

Addressable I/O Bus

Features - Two cycle skip line, program interrupt, teletype interface, console interface

Data Transfer Modes - Program controlled data transfers

Device Ports - A maximum of 50 physical ports shared between the data channels and the addressable I/O bus.

Options - API - Eight levels of automatic priority interrupts – Four hardware levels and four software levels

I/O PROCESSOR

The Input/Output Processor (See Figure 5-1) contains the control logic and registers necessary to transfer up to 18 bits of parallel data on a common bidirectional I/O bus. Data may be transferred directly between the I/O Processor and Memory, or between the I/O Processor and the accumulator (AC) of the CPU. All transfers are made on a request/grant basis, providing complete autonomy of processors and memory. The I/O Processor operates with a 1 μs cycle time. The processor accesses memory through the read-pause-write cycle which produces a synchronous memory cycle time of 1 μs. While

transfers are being made between Memory and the I/O Processor, the CPU is free to operate independently. Requests from the I/O Processor for memory access are, however, given priority over CPU requests by the memory multiplexer; this can cause the CPU an occasional "cycle-stealing" delay. The structure of the I/O Processor provides the following benefits to the user:

The simultaneity of data transfers and CPU computing permits high-speed processing to meet the demands of real-time applications.

The I/O Processor can be expanded or reconfigured at any time without major modification of the rest of the system.

User-designed or special-purpose equipment can be easily and inexpensively interfaced to the system.

Synchronous and asynchronous devices can be handled with equal ease.

Modes of Data Transfer

Peripheral devices may transfer data in any one of three modes: single-cycle block transfers, multi-cycle block transfers, and program-controlled transfers.

Data Channel Controller

The data channel controller implements the first two modes of data transfers and in addition has an add-to-memory mode and an increment memory mode. The real-time clock option is also implemented in this section.

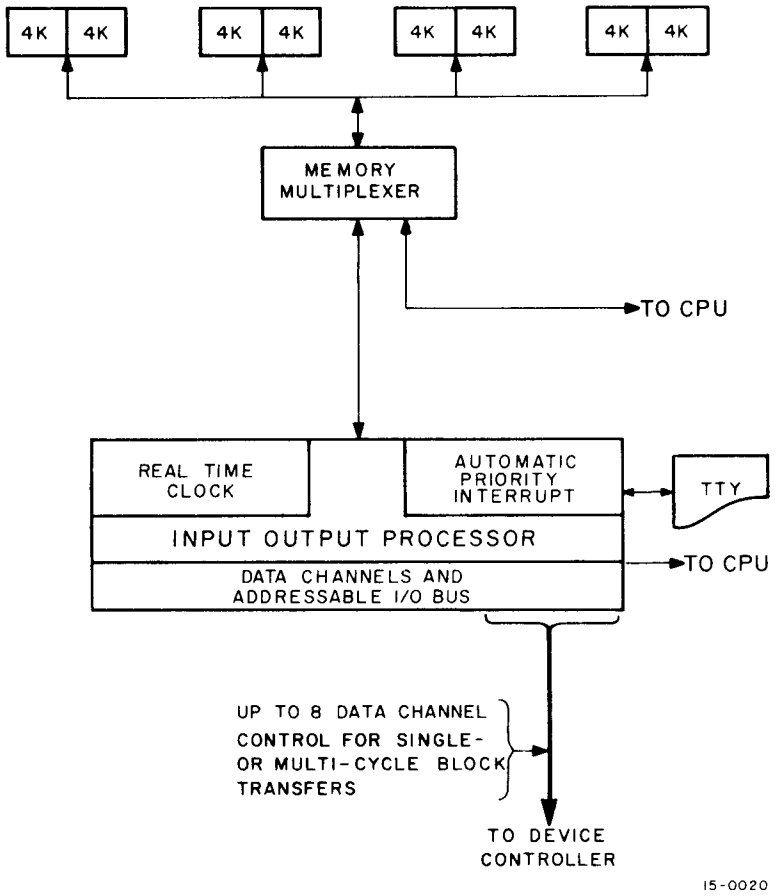


Figure 5-1. I/O Processor

Eight data channels are standard on all PDP-15 Systems and may serve to concurrently transfer data from eight different devices. Four of these are normally reserved for multi-cycle block transfers and the remaining four are reserved for single-cycle block transfers. However, the channels are designed to accept any mixture of either single- or multi-cycle devices.

Multi-Cycle Block Transfers

Normally, four of the eight standard channels are used for multi-cycle block transfers. A two-word packet in core memory is reserved for each of these channels: locations 30 and 31 for the first, 32 and 33 for the second, 34 and 35 for the third, and 36 and 37 for the fourth. The two words in the packet are used to store the "word count" (number of words to be transferred in the block), and the "current address" (where the data is to be transferred). The I/O Processor contains the control logic and an I/O adder to automatically fetch and increment the contents of the two registers.

Data is read into memory in three I/O Processor cycles and is read out in four cycles. (The additional cycle allows I/O bus settling and the settling of control gates prior to the strobing of the data word into the device buffer register.) Three memory cycles are required for both. Maximum input rate is 250,000 words/second and maximum output rate is 181,000 words/second, ensuring data transfer integrity.

A multi-cycle block transfer, flowcharted in Figure 5-2, is initiated by an input/output instruction after the two core registers have been initiated by minus the word count and the current address minus one. During the first cycle, the contents of the word-count register are incremented by one and restored. During the second cycle, the current address is incremented by one and restored, in addition to being transferred to the memory address buffer of memory. During the third cycle (or fourth in the case of output), the actual data transfer occurs. The I/O Processor continues to transfer data sequentially until the word-count register reaches zero, at which time an interrupt is generated to notify the monitor that the block transfer is complete. Because these multi-cycle block transfers are completely automatic in nature and do not require any CPU attention except for the I/O transfer initialization, the CPU is free to compute while they are taking place. The only limitation on simultaneity lies in the sharing of memory. The I/O Processor has first priority on memory requests and effectively "locks out" the CPU for three cycles. As data transfer rates approach maximum, the CPU can be completely locked out.

Figure 5-3 illustrates how the data channel controller registers implement the multi-cycle transfers.

Assume the two-word core-memory packets assigned to a given multi-cycle data channel have been loaded by the respective I/O service routine. For the case of data input to memory the following occurs:

An instruction from the service routine enables the device controller. This allows the controller to request a data transfer from the I/O processor.

When the device controller's data buffer registers are full it issues a "data channel request."

The I/O Processor, if not busy, acknowledges the request by returning a "data channel grant."

The device controller then generates a fixed code pointing to its packet address in core memory. This is transmitted over the common I/O bus address lines and is stored into the data storage register of the data channel controller through the I/O adder. The adder is inhibited during this operation.

The I/O Processor then generates a "memory cycle request."

The memory multiplexer, when ready, acknowledges by returning a "grant."

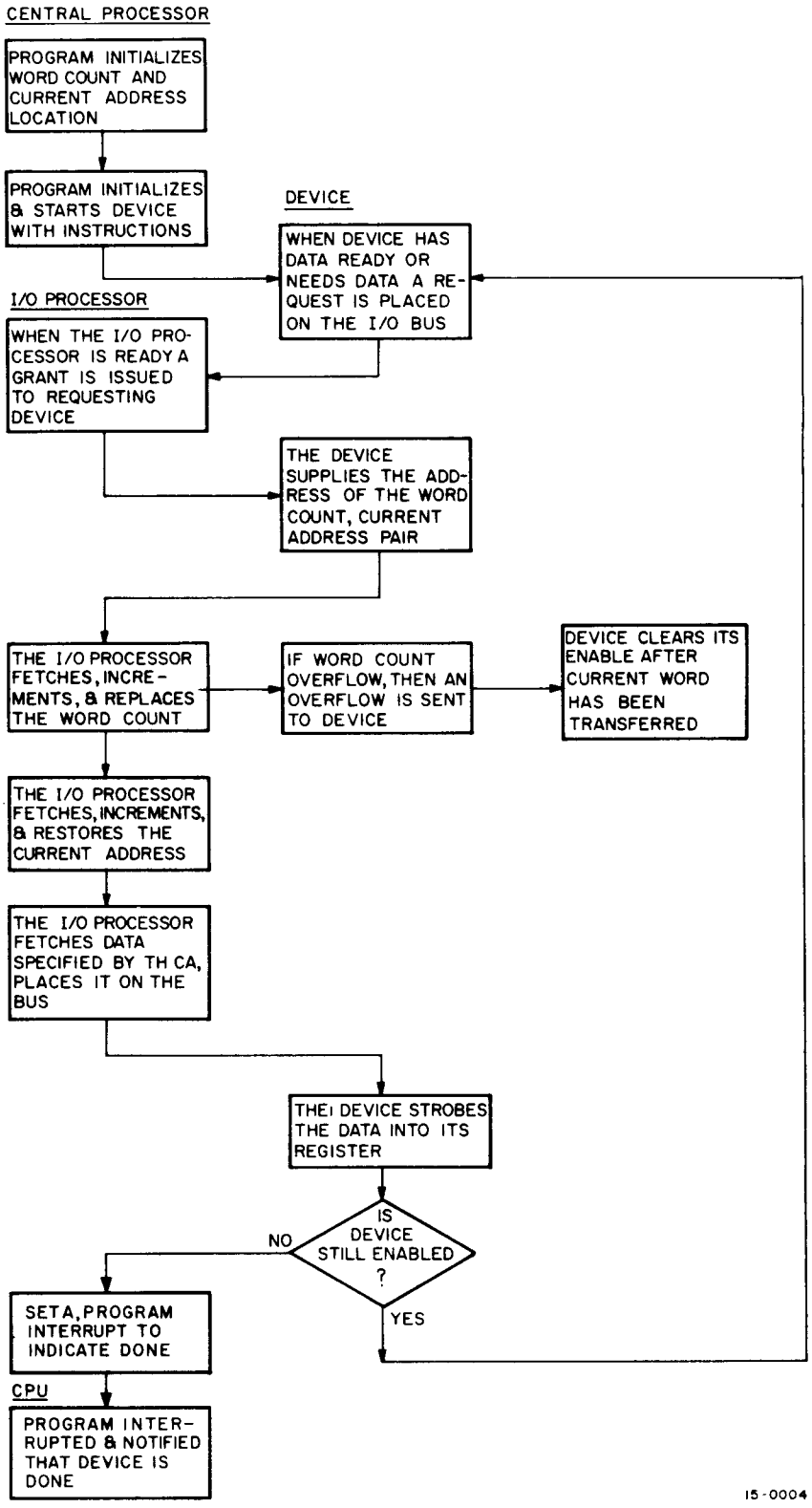


Figure 5-2. Multicycle Block Transfer, Flowchart

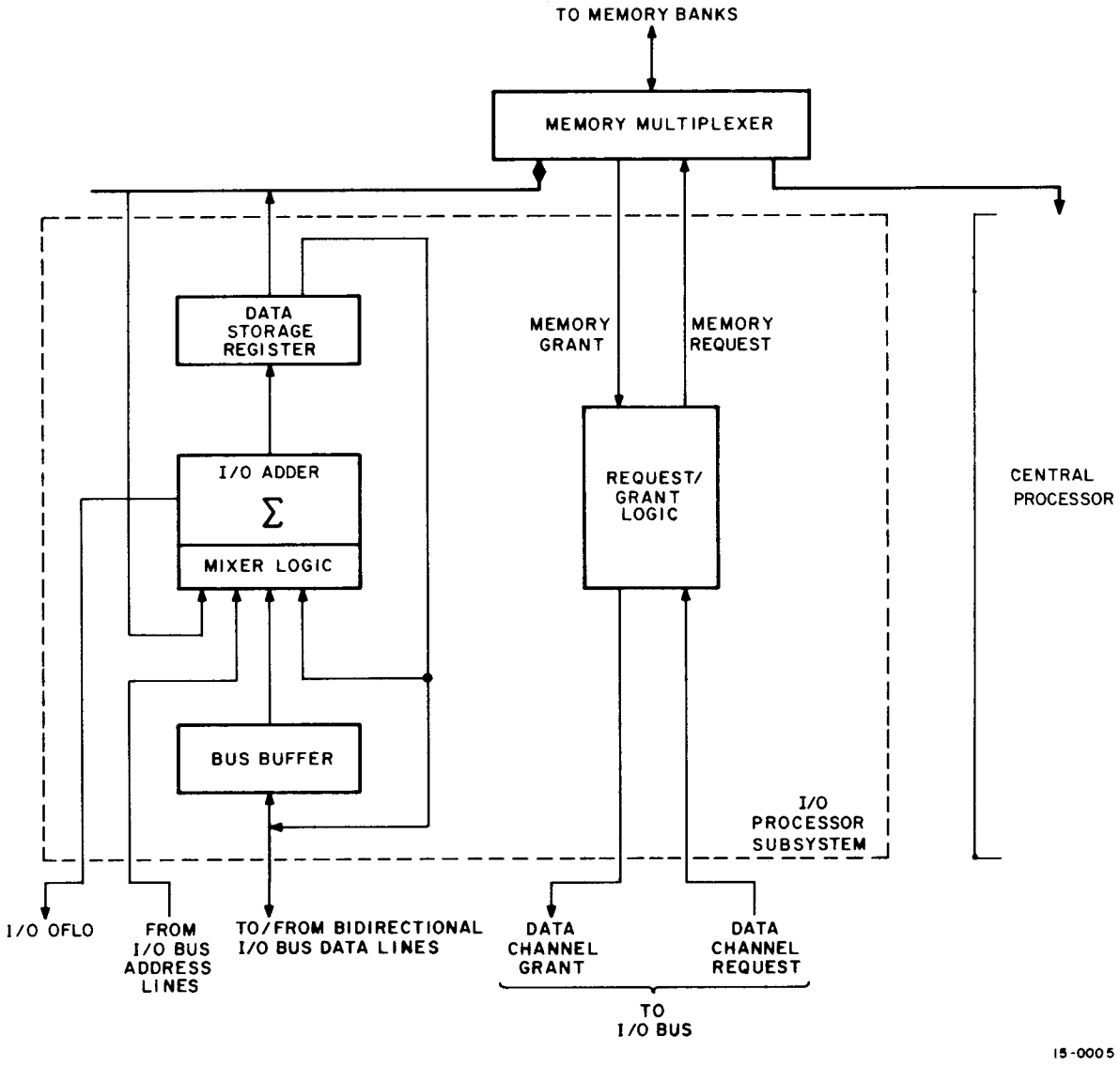


Figure 5-3. Multi-Cycle Transfer Implementation

The address data in the data storage register is then stored into the memory address (MA) register of the memory bank 1 and the data (word count) from the first word of the packet that the MA is now pointing to, is transmitted out of memory and into the data channel controller's adder. The word count data word is incremented by one and stored back into memory. If during this incrementing the adder overflows (indicating that the current address was the last), then an I/O overflow pulse is transmitted back to the device to disable future data-channel requests and also to post an interrupt to the Monitor.

This "word count" operation occurs in one I/O processor cycle, using one memory cycle.

During the second I/O Processor cycle, the fixed code from the device controller is gated through the adder and is incremented by one. It is then transmitted to the MA register to point to the second word in the

packet - the "current address," which is then transmitted back to the adder, incremented by one and strobed back to memory. During the third I/O processor cycle, the current address is strobed into the MA register to point to the data array word where the I/O data will be transferred. The data is then gated from the device controller, through the adder, (which is inhibited during this cycle), and into memory.

A memory request/grant synchronization again occurs; and

The data in the storage register is strobed into the data array ending the cycle.

Data output follows the same sequence, with the exception that one additional I/O processor cycle is required to allow the bus to settle before data is strobed out of memory and into the device.

Single-Cycle Block Transfers

Single-cycle block transfers, flowcharted in Figure 5-4, are used by high-speed peripherals that normally transfer complete records (blocks) of information, such as disks and CRT devices. A single cycle of the I/O Processor takes 1 μs allowing a maximum transfer rate of up to one million 18-bit words per second.

High-speed hardware registers, designed into the device controllers of the high-speed peripherals, store the "current address" (the memory cell where data is currently being transferred), and the "word count" (the number of words remaining to be transferred in a block). These registers are loaded by input/output transfer (IOT) instructions issued by the CPU. Device testing and initialization are handled by the CPU via IOTs to provide supervisory control. A subsequent IOT initiates the data transfer. The I/O Processor uses the current address information to address core memory, then strobes the data between memory and the device controller buffer register. Logic within the device controller then increments the current address register and the word count register to provide sequential block transfer.

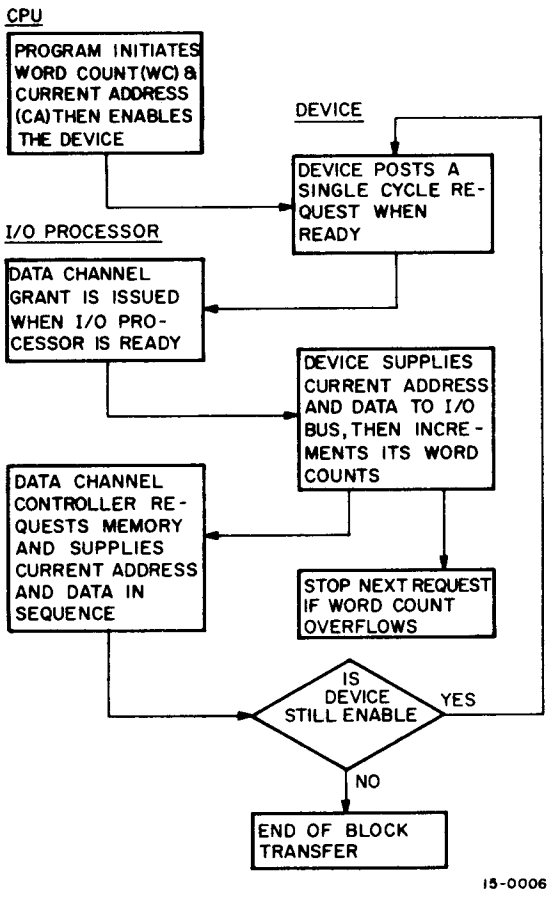


Figure 5-4. Single Cycle Block Transfer, Flowchart

When the word count register overflows at the end of a block transfer, an interrupt is generated to allow the monitor system to take further action. Typically, this action will include disconnecting the device from the I/O bus or reloading the device controller registers for another block transfer. The maximum number of words that may be transferred in a single-block is 32,768.

Figure 5-3 illustrates how the data-channel controller registers handle single-cycle transfers.

Assuming that the program has initiated the word count and current address of the device

controller, and has then enabled it, the following occurs:

The device controller posts a single-cycle data channel request to the I/O Processor.

The I/O Processor, as soon as it becomes available, acknowledges the request by returning a "Data Channel Grant."

The device then strobes both its current address and its data onto the I/O bus to the I/O Processor.

The data channel controller feeds the current address through its adder (which is inhibited throughout the single cycles) to the data storage register. A memory cycle is requested, and this address is strobed into a memory bank's address buffer. The data is then strobed off the 18 I/O data lines and into the memory location specified by the current address.

During this operation, the device increments its own word count, and disables itself on overflow. It then posts an interrupt to the monitor to indicate that its operation has been completed.

Increment Memory

The increment memory mode allows an external device to add one to the contents of any memory location in a single cycle; this feature is most commonly employed in the accumulation of data in histogram form. Effectively, the increment memory mode simply goes through the word-count cycle of a multi-cycle channel transfer, and then stops. The maximum rate at which it can increment is 500 kHz. This feature is particularly useful for in-core scaling and counting in pulse height analysis.

Add-To-Memory

Add-to-memory is a standard feature of the PDP-15 that adds unique capabilities to the already powerful I/O facilities.

In add-to-memory mode, the contents of an external register can be added to the contents of a memory location in four cycles. This feature is extremely valuable in signal averaging and other processes requiring successive sweeps for signal enhancement.

The add-to-memory operation is a combination of multi-cycle data channel input and multi-cycle data channel output operations. The data transmitted by the device is added to a word read out of memory as specified by the current address, and the result is rewritten into the same location. It is simultaneously transmitted to the device via the I/O bus. Four I/O processor cycles are required.

Real-Time Clock

When enabled, the real-time clock counts, in memory location 00007, the number of cycles completed by any one of three inputs:

a. The line voltage (50 or 60 Hz)

b. Any standard DEC clock may be optionally installed.

c. A user supplied TTL compatible signal which can be fed through a coaxial cable to a BNC connector on the PDP-15.

When location 00007 overflows, an internal program interrupt (or API request, if available) is generated informing the monitor that its preset interval is over. The monitor must either disable the clock or reinitialize location 00007 to the 2's complement of the number of counts it needs to tally.

The incrementing of location 00007 during a real-time clock request occurs via the I/O processors' increment memory facility. A real-time clock request takes priority over API, PI and IOT requests, but not over block transfers.

ADDRESSABLE I/O BUS

The addressable I/O bus implements the program-controlled transfers. It also contains the

program interrupt and the automatic priority interrupt (API) option.

Program-Controlled Transfer

Program-controlled transfers, implemented by input/output transfer (IOT) instructions, can move up to 18 bits of data between a selected device and the accumulator (AC) in the CPU. The devices involved are connected to the addressable I/O bus portion of the I/O processor. A total of up to 50 device controllers may be attached to this bus and to the data channels. IOT instructions are microcoded to effect response only for a particular device. The microcoding includes the issuing of both a unique device selection code and the appropriate processor-generated input/output pulses to initiate a specific operation. For an "out" transfer, the program reads a data word from memory into the AC. A subsequent IOT instruction places the data on the bus, selects the device, and transfers the data to the device. For an "in" transfer, the process is reversed; an IOT instruction selects the device and transfers data into the AC. A subsequent instruction in the program transfers the word from the AC to memory. Maximum transfer rate in this mode is 100,000 words per second.

As previously mentioned, IOT instructions are also used to initialize the single- and multi-cycle channels and the transfer word count and current address information to the single-cycle device controllers. In addition, these instructions are used to test or clear device flags, select modes of device operation, and control a number of processor operations.

A PDP-15 IOT instruction, Figure 5-5, contains the following information:

a. An operation code of 70_8.

b. An 8-bit device selection code to discriminate between up to 128 user peripheral devices (selection logic in a device's I/O bus interface responds only to its preassigned code). In normal practice, bits 6 through 11 perform the primary device discrimination between up to 42 devices, with bits 12 and 13 coded to select an operational mode or subdevice.

c. A command code (bits 14 through 17) capable of being microprogrammed to clear the AC and issue up to three pulses via the I/O bus.

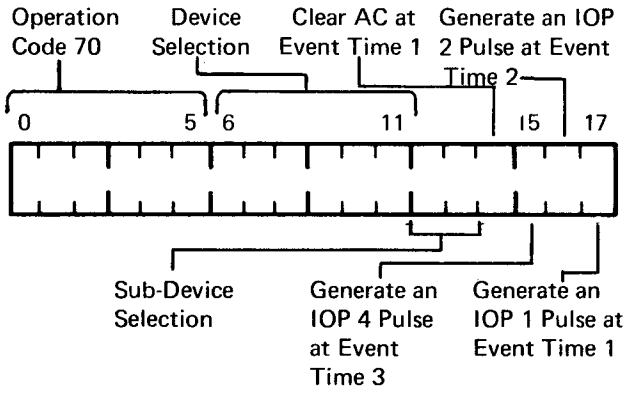


Figure 5-5. IOT Instruction Format

The four machine cycles required to execute an IOT instruction consist of the IOT fetch from core memory (memory is not accessed thereafter until completion of the IOT), and three sequential cycles each of 1 μs duration designated event times 1, 2, 3 (IOP1 through 4) (Figure 5-6). In IOT skip instructions, however, only IOP1 is used. These are two-cycle instructions. Bits 14 and 17 can be coded to initiate clearing of the AC and generation of an IOP1, respectively, during event time 1. Bits 16 and 15 can be coded to initiate generation of an IOP2 and IOP4 pulse during event times 2 and 3, respectively. IOT skip instructions are microprogrammed to produce an IOP1 pulse for testing a device status flag. IOP2 pulses are normally used to effect programmed transfers of information from a device to the processor. Because the AC serves as the data register for both "in" and "out" transfers, the "clear AC" microinstruction (bit 14) is usually microprogrammed with the IOP2 microinstruction; this combination effects clearing the AC during event time 1. IOP4 pulses

are normally used to effect programmed transfers of information from the AC to a selected device. These conventions do not, however, preclude use of the IOP pulses to effect other external functions if the following restrictions are observed.

The *usual* uses of IOPs are:

IOP1 - Normally used in an I/O skip instruction to test a device flag. May be used as a command pulse, but may not be used to initiate either a "load of" or a "read from" a device.

IOP2 - Usually used to transfer data to or from the device to the computer, or to clear device's information register. May not be used to determine a "skip" condition.

IOP4 - Used only to transfer data from the computer to the device. May not be used to determine a "skip" condition or to transfer data to or from the computer.

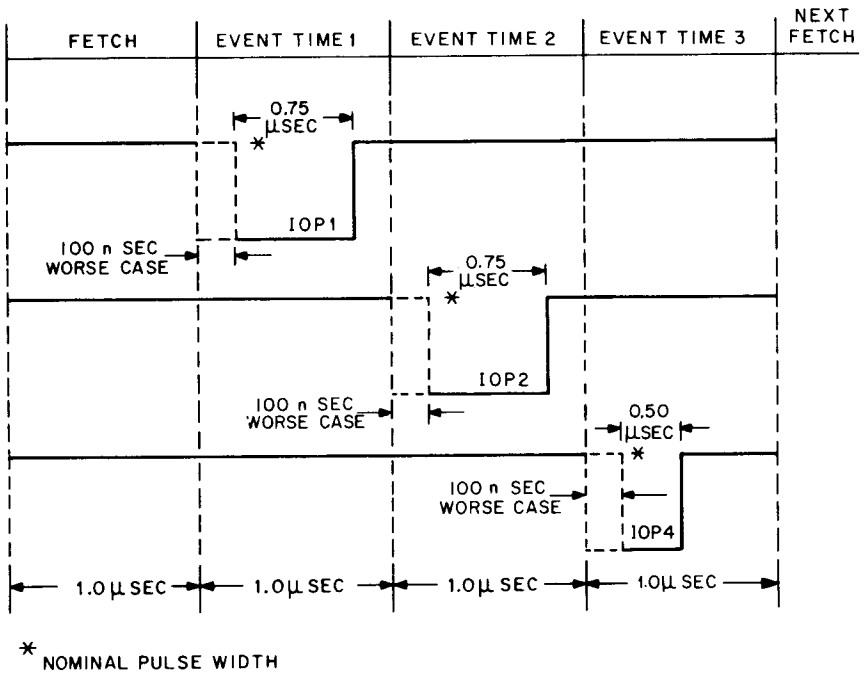


Figure 5-6. Machine Cycles for IOT Instruction Execution

PROGRAM INTERRUPT FACILITY

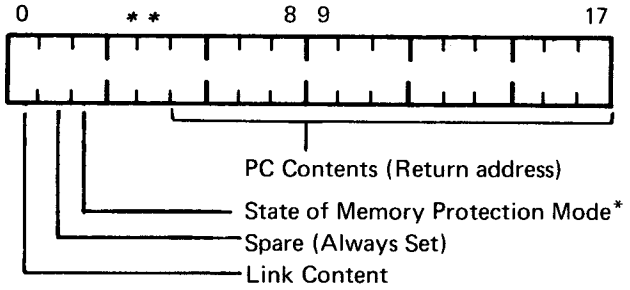
The program interrupt (PI) system, standard on all PDP-15 Systems, provides for servicing a peripheral device at rates up to 50 kHz.

The program interrupt (PI) facility, when enabled, relieves the main program of the need for repeated flag checks by allowing the ready status of I/O device flags to automatically cause a program interrupt. The CPU can continue with execution of a program until a previously selected device signals that it is ready to transfer data. At that time, the program in process is interrupted and the contents of the program counter (15 bits), memory protect mode (1 bit) and the link bit (1 bit) is stored in location 000000. The instruction in location 000001 is then executed,

transferring control to an I/O service routine for IOT instructions (see Figure 5-7). When completed, the routine restores the system to the status prior to the interrupt, allowing the interrupted program segment to continue. Where multiple peripherals are connected to the PI, a search routine containing device-status testing (skipping) instructions must be added to determine which device initiated the interrupt request. The program interrupt (PI) control is enabled or disabled by programmed instructions. When disabled the PI ignores all service requests, but such requests normally remain on-line and are answered when the PI is again enabled.

Mnemonic	Octal Code	Function
ION	700042	Enable the PI
IOF	700002	Disable the PI

The PI is automatically disabled when an interrupt is granted or when the I/O RESET key (on the console) is depressed. The PI is temporarily inhibited while the automatic priority interrupt system is processing a priority interrupt request. The PIE indicator (on the console) is lighted while the PI is enabled.



*Zero if respective option is not present.

Figure 5-7. Program Interrupt, JMS Instruction, or CAL Instruction Storage Word Format*

Conditional Skip-On Device Status

The PDP-15 order code includes a group of instructions for testing the status of peripherals. Instructions of this type direct the processor to skip the next instruction if the tested condition is true.

This group of instructions allows the testing of peripheral devices at the programmer's option. Normally rather than tying the processor up in a "wait" loop, the device signals that its buffer is ready by generating an interrupt. If it is a program interrupt, the "conditional skip" is used in a so called "skip chain" to find which device initiated the interrupt. Each skip instruction takes 2 μs.

AUTOMATIC PRIORITY INTERRUPT

The automatic priority interrupt (API) system, standard with the PDP-15/30 and 15/40 Systems and optional with the 15/10 and 15/20 Systems, ensures efficient handling of service requests (without any loss of data) at high rates.

The API system contains eight levels of priority. The lower four of these are allocated to the monitor systems, the upper four to the I/O Processor. Up to 28 individual interrupts are available at the I/O processor. They share the four levels of priority, with the restriction that not more than eight may be on any given level.

A device initiates an interrupt request on its preassigned level by raising a request flag, and identifies itself by posting a unique core address. There is one unique core address for each of the 28 interrupts (44_8 through 77_8). This address serves as the entry point (trap address) to the device's service routine.

Each monitor API level services one interrupt and uses a single trap address between locations 40_8 and 43_8. The monitor requests are initiated by a program issuing an ISA instruction.

The I/O interrupts permit the asynchronous operation of many devices, each at its proper priority level. The software priority levels are used to establish a priority queue for the processing of real-time data without inhibiting the hardware interrupts to service devices.

API Hardware

Figure 5-8 relates the activity of the Automatic Priority Interrupt system from the initiation and

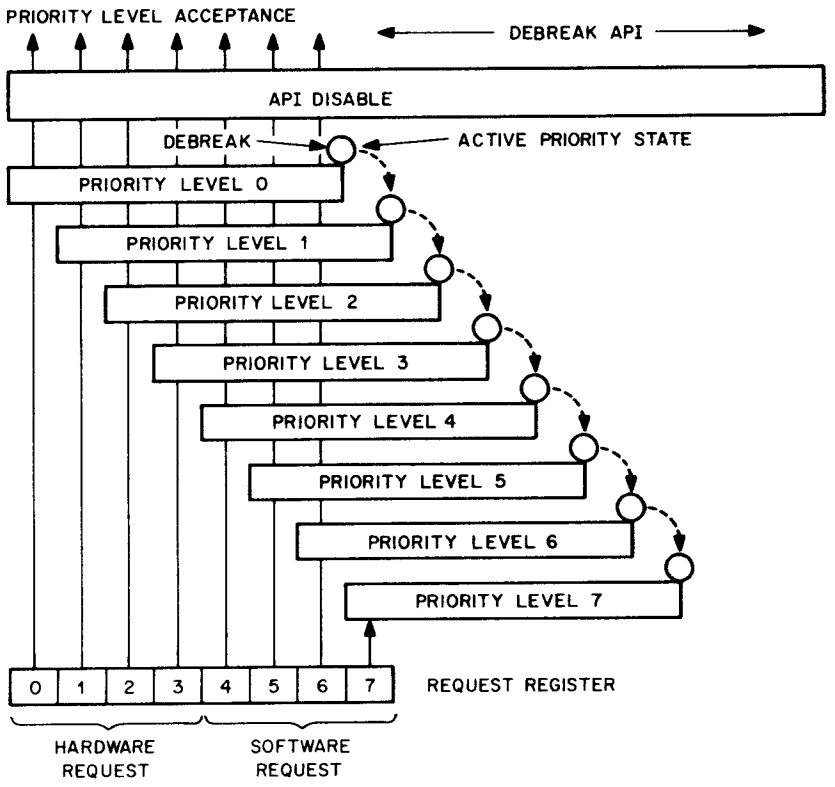


Figure 5-8. API System, Simplified Block Diagram

acceptance of the request, the servicing of the accepted request and the debreak from the serviced priority level.

The request register contains eight levels; four levels are activated by the devices (hardware) on the I/O bus, and four are activated under software supervision. The hardware requests are assigned the highest priority, and are demonstrated as requests 0, 1, 2, and 3. The software requests are demonstrated by requests 4, 5, 6, and 7, and are initialized by the ISA instruction with the associated AC bits set.

The priority level *bars* depict the priority level that is selected by the ISA instruction, or that is raised by the API control when it has granted a request on that specific level. The priority PL bars indicate that any request equal to, or less than (in priority), to the priority level selected will *not* be accepted. At the end of the subroutine currently being performed by an active request, a debreak and restore instruction is issued to lower the priority to the next *selected* priority level. The ball, representing the priority debreaking, will fall as long as there is *no* bar present (i.e., no priority level set). If a lower priority level is set, the debreaking will cease at *that* level.

The API request register (RR) buffers the inputs from the hardware interrupt on levels 0 through 3 and the inputs from the monitors on levels 4 through 7. Up to eight interrupts may be attached to a single level. If two or more of these make simultaneous interrupt requests, the interrupt closest to the processor is given priority. An interrupt request sets a bit in the RR according to its preassigned priority level. When the scanner detects that bit, the API system signals the CPU to stop execution at the completion of its current instruction. It then gates the I/O processor's 15 address lines, which contain the address of the interrupt's unique core location, into the CPU memory output register. The CPU then requests a memory cycle and executes the instruction it fetched from that location. During this operation the program counter remains unchanged. The instruction is normally a jump to

subroutine JMS which stores the contents of the program counter, which points to the location where the current program was interrupted, in the first location of the subroutine and begins execution at the subroutine's second location. The API system also sets a bit in the PL corresponding to the level of the interrupt. This prevents interrupts on the same level or lower levels from interrupting the current interrupt. The scanner continues to sample the higher levels so that higher priority devices can interrupt lower priority devices. The JMS instruction allows nesting of all levels. At the completion of the interrupt subroutine, a debreak and restore (DBR) instruction must be issued to reset the bit in the PL and in the RR.

The API hardware ensures that simultaneous requests by multiple devices are handled in the proper priority sequence. If interrupt requests occur at different priority levels, the highest priority requests will be serviced first. Higher priority devices may interrupt lower priority devices. The entire API system may be enabled or disabled with a single instruction; however, most devices provide facilities to connect and disconnect their flags from the interrupt separately. If the API system is disabled, the device will automatically signal the program interrupt to obtain a response at that priority level.

Under program control, the level of a priority request may be raised to provide dynamic priority reallocation. It does this by issuing an ISA.

ISA	705504	Initiate selected activity. The API activity specified by a set bit in the AC is initiated (refer to instruction set Chapter 7).

The ISA instruction places a bit into a priority level specified. This effectively masks all lower priority levels. A debreak instruction (DBK) is used to reset this bit when the higher priority level is no longer required. For example: a priority-2 interrupt routine is designed to enter data in memory locations A through A + 10 during an interim period when the priority-2 device is inactive and, based on a calculation made by a software priority-6 routine, it becomes necessary to move the data to memory locations B through B + 20. The changes in the routine at level 2 must be completed without interrupt once they are started. This is possible by causing the level-6 program to raise itself to level-2 (devices on the same or lower priority may not interrupt), complete the change, and debreak back to level-6. Note that the ISA is also used to trigger the API levels dedicated to software priority queues. This unique advantage of this API system lies in the proper use of its software levels. In real-time environment, it is necessary to maintain data input/output flow, but it is not possible to perform long complex calculations at priority levels which shut out these data transfers. With the API, a high-priority data input routine which recognizes the need for complex calculations can call for a software-level interrupt. Since the calculation is performed at a lower priority than the device handling, the latter can go on undisturbed. The monitor task of establishing a multi-priority request queue at the software level is greatly simplified by utilization of the API hardware.

COMMON I/O BUS

The I/O Processor contains a common I/O bus (see Figure 5-9) to transfer both data and IOT instructions to the block transfer channels and to the addressable I/O bus. The bus is the major communication path for I/O devices. It consists of cables which link all the I/O device controllers to a common interface point at the I/O Processor. All signal lines for command and data transmission arising from the data channels, addressable I/O bus, operation of the multi-level automatic priority interrupt system, program interrupt, I/O status read, and I/O skip facilities, are contained on this bus. The bus length can be up to 75 ft.

Data Lines

Eighteen data lines constitute the bidirectional facility for transferring data bytes of up to 18

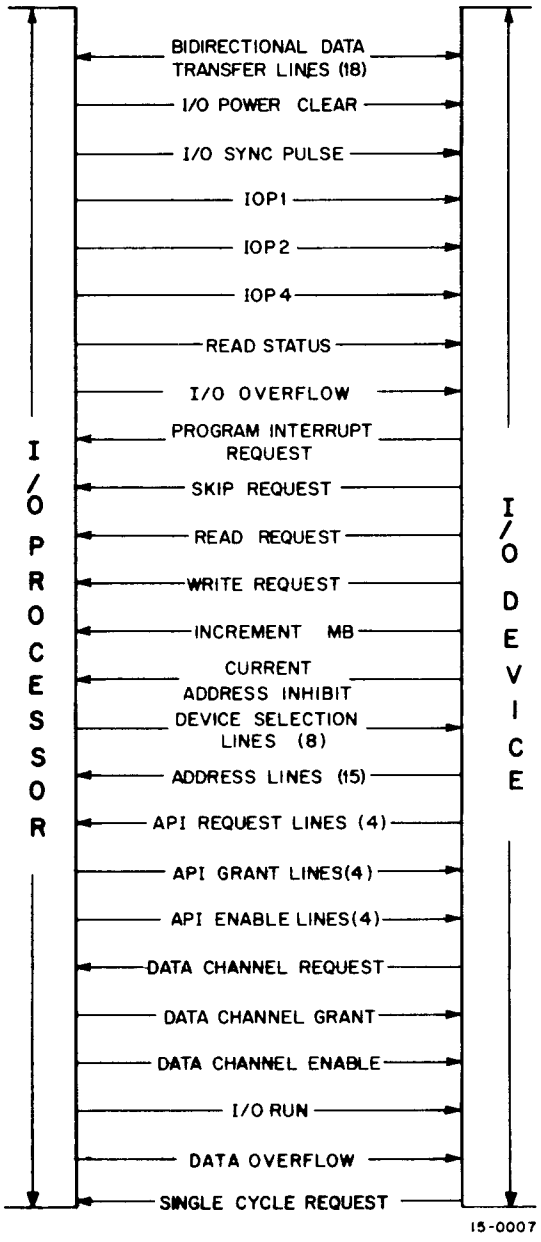


Figure 5-9. Common I/O Bus

bits between the I/O Processor and all I/O devices. Transfers are made on a dc basis with the processor or device allowing bus settling time before data on the lines is strobed into the receiving register. The data lines convey data between the memory buffer register and a selected device buffer register for block-transfer channel operation; they transfer data between the accumulator and a selected device buffer register for program-controlled transfers.

Output Control Signals

Eight output control signals are generated to effect specific functions in a selected device.

I/O Power Clear

The I/O power clear signal resets all flip-flops storing device-to-processor flag indications (e.g., ready, done, busy). It is issued by power turn-on, the occurrence of a clear-all-I/O flags (CAF) instruction, and by actuation of the I/O RESET key on the control console. This pulse is also used (in conjunction with the device select lines) to initiate automatic read-in from the selected device.

I/O Sync

I/O sync may be used to synchronize I/O device control timing to the processor. It is issued every I/O processor cycle.

IOP1, IOP2, IOP4

Microprogrammable signals are used to effect IOT instruction-specified operation within a selected I/O device. The I/O Processor automatically generates IOP2 or IOP4 for input or output transfers. Although they may be used for other control functions, the common uses of the IOPs are:

IOP1 - to test a device flag (in an I/O skip instruction). It may be used as a command pulse, but it cannot be used to initiate loading of, or reading from, a device buffer register.

IOP2 - to transfer data from a selected device to the processor, or to clear a device register.

IOP4 - to transfer data from the processor to a selected device register. It may not be used to determine a skip condition or to transfer data to the processor.

Read Status

Read status is issued by execution of the input/output read status (IORS) instruction. It loads the AC with an 18-bit word containing device flag indications for devices interfaced to the read-status facility.

Input/Output Read Status

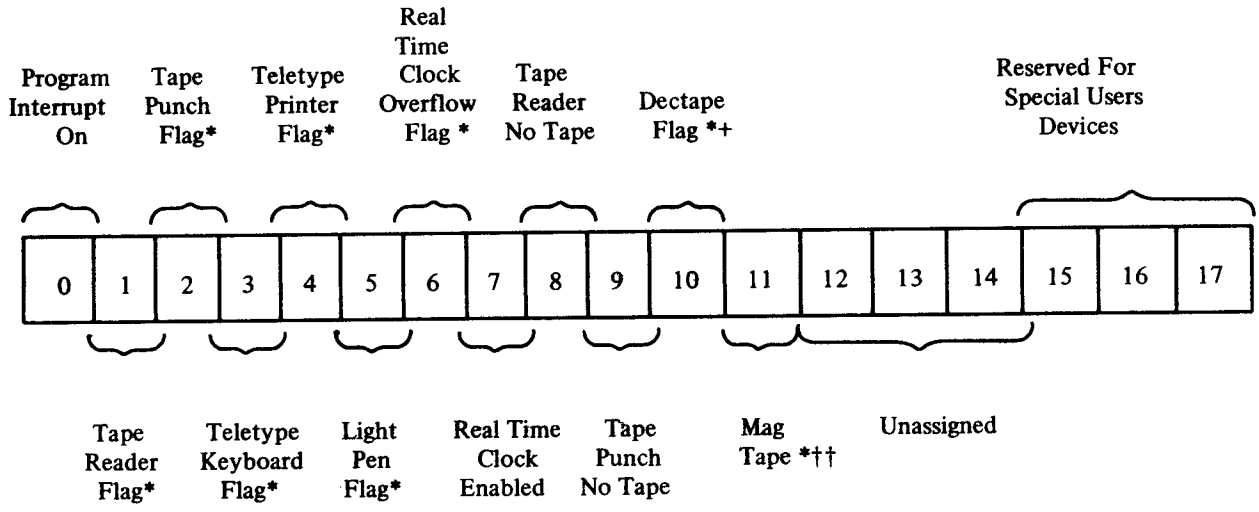
The input/output read status facility provides for programmed interrogation of the status of those external devices using this facility. Upon execution of an input/output read status (IORS) instruction, the states of those device flags (done, busy, not ready, etc.) interfaced to this facility by the I/O bus are transferred to specific assigned bit positions of the AC. This allows the program to check for specific flag conditions or display the flag states on the operator's control console. Figure 5-10 shows the bit positions associated with the commonly interfaced flags.

The IORS word can contain up to 18 flag bits. Those bits not used are zeroed. The presence of a flag is indicated by a 1 state in the corresponding AC bit.

Switching the REGISTER DISPLAY control (on the console) to the STATUS position simulates execution of the IORS instruction (with the processor in the "program stop" condition). The contents of the IORS word (i.e., the states of the device flags) are displayed in the REGISTER indicators (on the console) at this time.

I/O Overflow

I/O overflow is issued during the first cycle of the block-transfer operation if the contents (2's complement) of the word counter assigned to the currently active channel device becomes 0 when incremented. This indicates that the program-specified number of words will have been transferred at completion of the channel transfer



* Will cause a program interrupt
\+ Inclusive or of transfer completion and error Flags
*†† Inclusive or of MTF and EF

Figure 5-10. I/O Read Status Bit Assignments

in progress. It is normally used to turn off the device, thereby preventing further channel action by that device until a service subroutine reinitializes the channel's word-counter and current-address registers, and the program turns on the device request flag. The overflow signal may also be used to initiate a program interrupt through the program interrupt or automatic priority interrupt facility for access to the initializing subroutine. Additionally, I/O overflow occurs when an I/O increment memory operation causes the location to overflow.

Data Overflow

Data overflow is issued during the third cycle of a four-cycle add-to-memory data transfer when the addition operation overflows.

Input Control Levels

Six input control level signals arrive at the I/O processor:

Program Interrupt Request - A device delivers this signal to request interruption of the program in progress. The program traps to location 00000 when no I/O transfer action of higher priority is in progress. The instruction resident in location 00001 is fetched and executed. If more than one device is connected to the program interrupt, this instruction transfers control to a subroutine which determines through a search process (skip chain) the device making the program-interrupt request. The appropriate service routine is then accessed.

Skip Request - The return of this signal to the processor indicates that an IOT instruction test for a skip condition in a selected device has been satisfied (e.g., a test of ready status). The program counter is subsequently incremented by one to effect a skip of the next instruction of the program in progress.

Read Request - This signal requests that the processor execute a data-channel-read transfer of a data word into the processor.

Write Request - This signal requests that the processor execute a data-channel-write transfer of a data word into the selected device's information register.

MB Increment - This level requests that the processor increment (by one) the contents of the memory location address specified by the 15-bit address on the I/O bus address lines.

Current Address Inhibit - This is a special signal line required by devices which automatically search for records, etc; typical are DECtape and magnetic tape. The presence of this signal level inhibits normal incrementing of the device-assigned current address register during a data channel transfer.

Device Selection Levels - Identification of the current instruction as an IOT causes the bit pattern placed in the CPU MI 6 through 13 at the fetch of the instruction to be bus-driven and sent via eight bus lines to device selection modules contained in the control logic for each device. These eight levels form a device selection code and 2-bit subdevice or mode-selection code.

Address Lines

Fifteen lines constitute an input bus for the devices which must deliver address data to the processor. There are two uses for the address bus:

a. When a device interfaced to the multi-level automatic priority interrupt system receives an I/O processor grant of its interrupt request, it delivers to the CPU a hardware-defined address, relating to its API channel assignment. This channel address indicates to the device's service routine the location of the unique transfer vector.

b. When a block-transfer channel device receives a processor grant of its transfer request, it delivers to memory a hardware-defined address indicating the memory location of the assigned channel's word-count register.

Multiplexed Control Lines

Fifteen control lines serve as processor-to-device-control information paths, three for the block transfer channel facility and twelve for the priority levels on which the automatic priority interrupt system processes requests for service. Control lines are used in the following ways:

Request - a device transmits a service request to the processor via the appropriate request line. There are four automatic priority interrupt request lines (one for each level) and two data channel request lines (single-cycle requests and multi-cycle requests).

Grant - the processor indicates a grant of the service request.

Enable - the enable signal controls the priority order for answering service requests of devices interfaced to the block transfer data channels or to one of the API's device channels. Priority for a channel (data or API) is allocated in descending order from the device nearest the processor I/O bus interface. An enable signal permits servicing of the requesting device with the highest channel of priority and inhibits all lower-priority devices from making requests during the interval of service.

I/O Run

The I/O Run signal is available at the interface for use as the interface designer requires. This bus driven level switches to the +5V level and remains there while the RUN flip-flop in the CPU is set. A ground level indicates that the RUN flip-flop has been cleared.

Teletype Interface and Hardware Read-In

The I/O Processor includes a Teletype control and Teletype unit as standard input/output equipment. Teletype Models Types 33 and 35, KSR or ASR, will operate with this controller.

The Teletype is capable of inputting and outputting at a rate of 10 characters per second. Serial transmission and reception of an 8-level character code is over a 4-wire cable connecting the Teletype and the control, which is located in the I/O Processor.

For the ASR units the reader and keyboard are electrically tied together and the punch and printer are mechanically connected. Teletype input functions are logically separated and therefore the keyboard/reader and printer/punch may be considered as individual devices. The program must echo any character from the keyboard it wants printed.

Hardware controlled read-in facilities are provided for reading paper tape into memory via either the ASR or the Type PC15 High-Speed Reader. This method of reading paper tape is accomplished by placing the tape into the reader and pressing the READ-IN key on the console. A hardware program wired into the I/O Processor will read the tape into memory. Control is then automatically transferred to the beginning of the bootstrap program which initiates execution.

Priority

In view of the autonomous substructures of the PDP-15, three types of priority must be considered: memory access priority, priority on the common I/O bus, and priority on the use of the CPU.

Memory Access - The I/O Processor always has priority over the CPU in accessing memory. However, once a CPU memory request has been granted it will be allowed to complete it cycle (800 ns) before control can be returned to the I/O Processor.

Common I/O Bus - Prioirity on the I/O bus is of concern only when more than one device is transferring information on the I/O bus and the I/O bus requests are received by the I/O Processor. The following order of priority occurs:

a. Block Data Transfer Channels - The eight block data transfer channels range in priority from channel 1, which has the highest priority to channel 8, which has the

lowest. Normally, single-cycle block transfer devices are placed on the high-priority channels to give them preference over the multi-cycle devices. However, if the data transfer rate of a single-cycle device is not critical, it can be placed on a lower priority channel to give preference to both single- and multi-cycle devices. For example, a display processor might be placed on a low-priority channel, since a temporary delay of data during the refresh cycle is not critical.

b. Real-Time Clock - The real-time clock has priority after the block data transfer channels. The real-time clock utilizes the I/O processor to fetch the contents of a reserve core memory cell (000007_8), increment the count, and then restore the new count.

c. Automatic Priority Interrupt - The automatic priority (API) system adds eight additional levels of priority to the PDP-15. The upper four levels are assigned to devices and are initiated by flags (interrupt requests) from these attached devices. The lower four levels are assigned to the programming system and are initiated by software requests. The priority network insures that high data rate or critical devices will always interrupt slower device service routines while holding still lower priority-interrupt requests off-line until they can be serviced. The API identifies the source of the interrupt directly, eliminating the need for a service routine to flag-search.

d. Program Interrupt - The program interrupt (PI) facility offers an efficient method of I/O servicing, if the API system is not used. The computer continues with execution of a program until a previously selected peripheral device signals that it is ready. At that time, the program in process interrupts and transfers control to a service subroutine. When completed, the subroutine restores the computer to the status existing prior to the interrupt, allowing the interrupted program segment to continue. Where multiple peripherals are connected to the PI, a search routine with device-status testing (skipping) instructions must be added to determine which device initiated the interrupt request.

e. Program-Controlled Transfers at the Main Program Level.

CPU – Priority on the use of the CPU is established by the API level, the program interrupt, and the main program, in that order.

Latency

I/O transfer latency is a measure of the time between a device's request for service and the actual performance of that service. Regardless of a device's priority, once its request for service has been granted, the I/O Processor holds all other devices off until the current service request is complete. For example, a single-cycle device on data channel 1 requesting service just after the initiation of a multi-cycle out-transfer would have to wait for four I/O processor cycles before using the I/O bus. If a multi-cycle transfer request for memory comes just after a grant to the CPU, an additional 800 ns are added to the latency. Finally, clock synchronization can take additional time resulting in a worst case latency of less than 6 μs for the requesting single cycle device. The worst-case latency for each peripheral (see Table 5-1) on a PDP-15/40 System (shown in Figure 5-11) with a 1000-line-per-minute printer, PR15 disk pack and CR03B 200-card-per-minute card reader, is shown in the following table. Devices are listed in the order of priority in which they are serviced.

With all of these devices active and transferring data concurrently, less than 50% of the I/O processor capacity is utilized. The remainder is available for real-time devices such as A/D and D/A converters as well as for graphic terminal devices and other peripherals.

Table 5-1
Worst Case Latency, PDP-15 Peripherals

Device	Maximum Transfer Rate	Allowable Latency (in μs)	Worse-Case Latency in this system (in μs)
RP15 Disk Pack	130,000* word/second	14	7
RS15 DECdisk	62,000 words/second	16	12
TU55 DECtape	5,000 words/second	200	17
LP15 Line Printer	1,000 lines/minute	40**	36
Real-Time Clock	1,000 cycles/second	1,000	61
PC15 Paper Tape System	300 characters/ second (reader)	3,333	62 + subr
CR03 Card Reader	200 cards/ minute	3,750	190
KSR 35 Teletype	10 character/ second	100,000	250

*The RP15 is double-buffered with two 36-bit registers. Two 18-bit words are transferred back-to-back on the I/O bus.

**The LP15 transfers two 18-bit words every 40 μs.

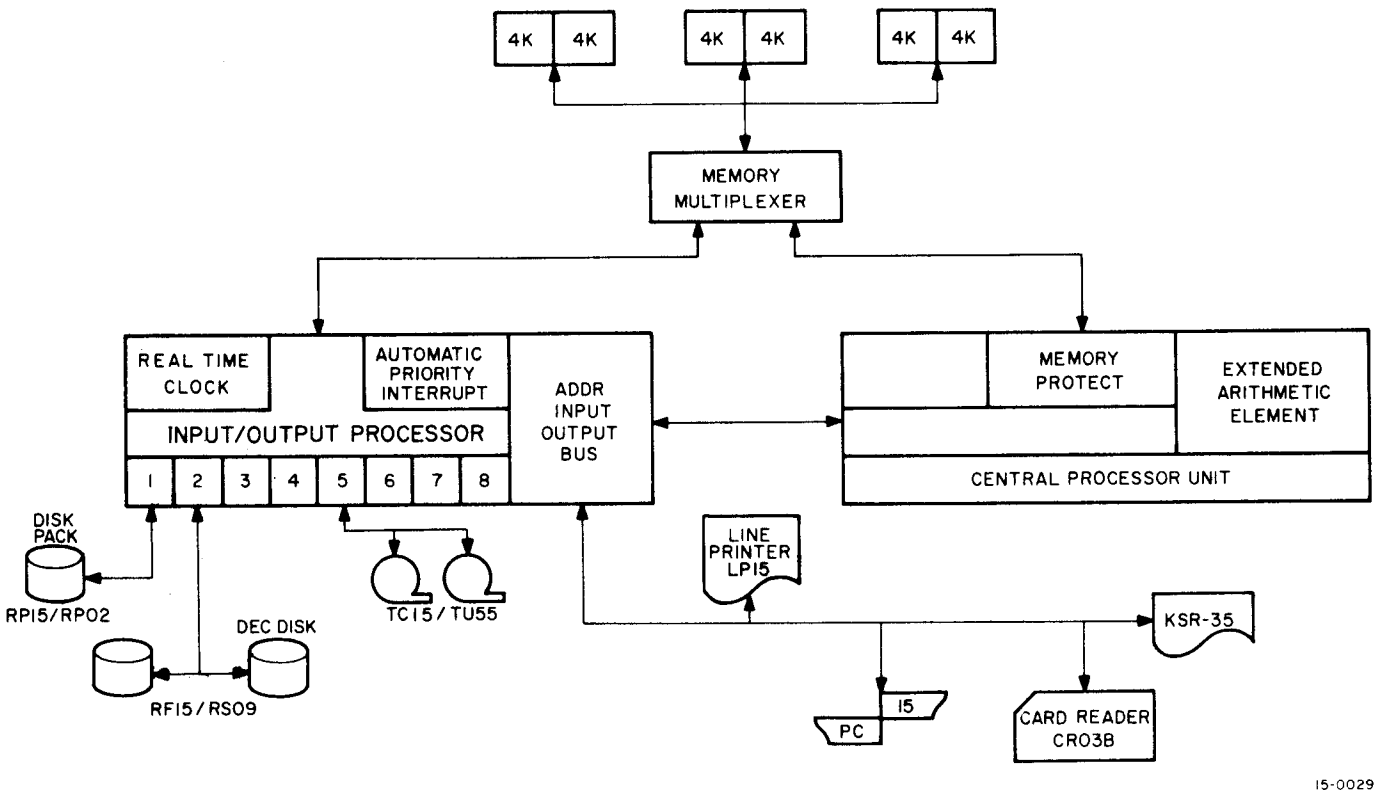


Figure 5-11. Expanded PDP-15/40 System

CHAPTER 6
ADDRESSING

This chapter describes the PDP-15 addressing scheme and the basic data word formats.

INTERPRETATION OF WORDS FROM MEMORY

Words stored in magnetic core memory are strings of 18 bits. An instruction word is indistinguishable from a data word. The Central Processor which decodes and implements instruction words, differentiates data words from instruction words by the sequence in which they are retrieved from storage. The program counter is used to point at the location of the next instruction. The instruction itself, if a memory reference instruction, then points to the memory location where data is to be fetched or stored. There are two types of instruction words:

a. Those which reference memory by indicating in the operand address field the location of the data necessary to carry out the operation (e.g., Add the contents of memory location 1000 to the accumulator).

b. Those that deal with control and do not require a memory reference cycle (e.g., Shift the contents of the accumulator two binary bits to the right).

INFORMATION RETRIEVAL FROM MEMORY

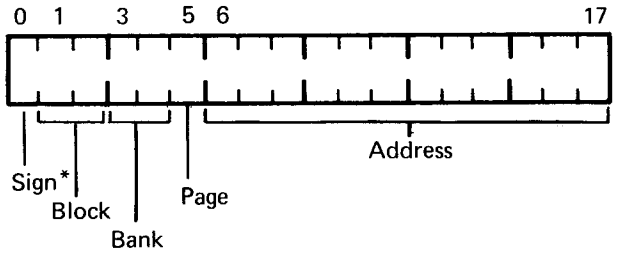
The basic concept involved in retrieving information from storage is that each piece of information has an address, similar in nature to a street address to locate a building, or a zip code to locate a postal zone. Integer addresses ranging from 0 to $32{,}767_{10}$ (or 00000_8 to 77777_8) are used by the PDP-15 to locate information in Memory. Figure 6-1 shows a diagram of the program counter register. Bits 3 through 17 are used to address the first 32,768 core locations in Memory.

Bits 3 and 4 select the memory bank being addressed (banks 1 through 4) and bit 5 determines which page (page 0 or page 1) of that bank the word is in. The remaining positions of the address are used to select one of the 4096 words on that page.

NOTE

The PDP-15 is designed to allow for special expansion to 131,072 words of Memory, so that address registers such as the program counter are 18 bits long.

The *Program Counter* is an 18-bit register which points to the next instruction. This register can be loaded from the console switches to begin execution of the program. At the beginning of each instruction, the program counter is incremented by one to specify the memory location of the next instruction. However, this incrementing is performed modulo 4096. When the PC is incremented, only the 12 low-order bits function as a 12-bit counter. There is no overflow into the high order bits.



To change pages, or banks within a 32K block, a jump indirect is normally used. To change blocks for systems greater than 32K, a jump indexed is used. These instructions replace the contents of the program counter with 15 bits or 17 bits, respectively, to effect both a new bank-page or 32K block address.

MEMORY REFERENCE INSTRUCTIONS

Figure 6-2 shows the format of PDP-15 Central Processor memory reference instructions. The bit positions in the 18-bit word are numbered from 0 to 17, counting to the right with bit 17 as the low-order bit. This convention will be employed throughout this manual. A PDP-15 word will be construed as an instruction word only when it has been retrieved from storage and transferred into appropriate registers in the Central Processor.

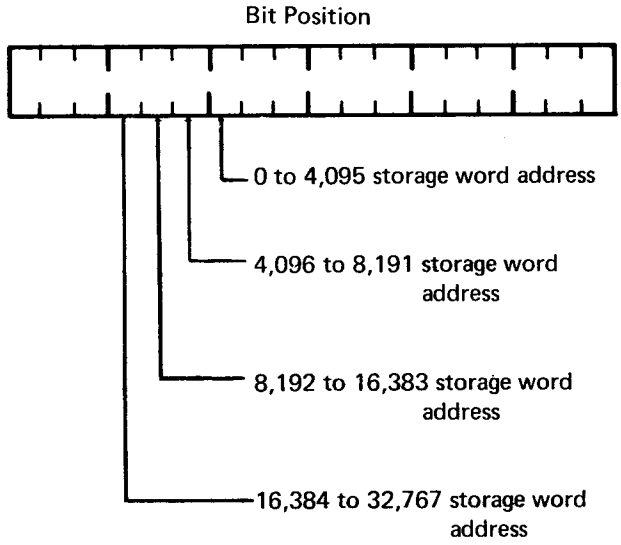


Figure 6-1. Program Counter Register

There are three fields in the memory reference instruction word; the operation code, the address mode (E Field) and the operand address field.

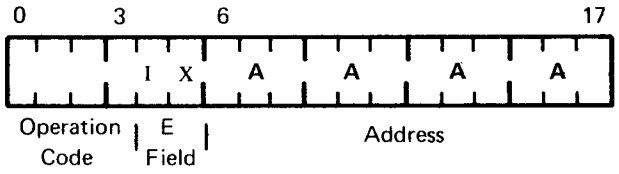


Figure 6-2. Memory Reference Instruction Format

Operation Code Field

The 4-bit operation code occupies bits 0 to 3 of the word and specifies 1 of the 13 memory reference class of instructions. The remaining three codes in the operation field are used to specify the non-memory reference instructions and are discussed later in this chapter.

*The sign bit is unused in address pointing since the PDP-15 System does not reference negative memory.

Operand Address Field

The 12-bit operand address field occupies bits 6 to 17. The number contained in the address field is the address of a word located in core storage (which is usually one of the operands of the instruction; e.g., ADD Y) although this address may be modified before it actually references a word in Memory. Since this field is 12 bits long, the instruction directly addresses 4096 words of Memory, or one full page.

Address Mode

The 2-bit address mode or E Field occupies bits 4 and 5. The 2-bit E Field indicates address modification as illustrated by the flowchart, Figure 6-3. There are four combinations as shown by the following:

E Field	Value	Operation	Addressable Memory	Δt*
0 0	0	Direct	4K	0
0 1	1	Indexed	128K	0
1 0	2	Indirect	32K	.8
1 1	3	Indirect-Indexed	128K	.8

*Additional time required for address modification.

E=0 No Address Modification - The 12 bits in the address field point directly to the operand's address in the current page where the data will either be fetched or stored. The address pointer is formed in the operand address register by concatenating the block, bank, and page address stored in the program counter to the 12-bit address field. Since the block, bank, or page address does not change unless a field change instruction is issued (jump indirect or indexed), the term current page address always infers the concatenation process.

E=1 Indexed Address Modification - The content of the index register (18 bits, including sign) is added to the current page address forming the effective address where data will be transferred. An indexed instruction can directly address any portion of core memory and does not add any additional time to an instruction's execution time.

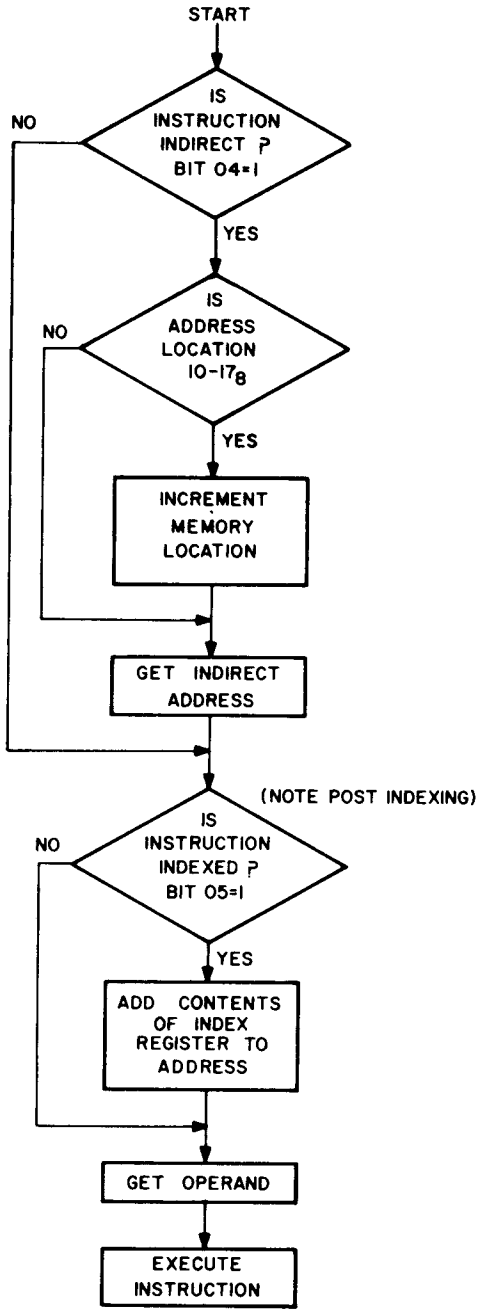
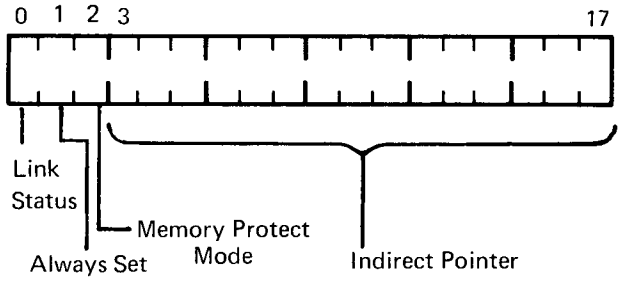


Figure 6-3. Address Modification Flowchart

E=2 Indirect Addressing - When indirect addressing is indicated, the current page address is

taken as containing not the operand but an indirect pointer to the location of the operand address.



The indirect pointer is concatenated with the current block address, bits 1 and 2 of the program counter, in the operand address register to form the effective address where data is transferred. A second memory reference cycle is then required for obtaining the actual data.

Only one level of indirect addressing is permitted. Fifteen bits of the indirect word are used as the address, permitting 32,768 possible words in a current block to be accessed. The first three bits in the indirect pointer are used to store status information and are not used for address formation.

E=3 Indirect and Indexed Address Modification - When both indirect and indexed addressing are indicated, the indirect operation occurs first as shown above. Then, the value of the index register is added to form the effective address.

Indexing Example - The following example shows a typical use of the index register.

A second register, called the limit register, (18 bits), is used to test the index register when it is used in a loop. An "Add to Index and Skip" (AXS) instruction causes a signed eight-bit number, contained in the last eight bits of the AXS instruction, to be added to the index register. The index register is then compared to the limit register and if the sum in the index register is greater than the limit register, the next instruction is skipped.

Form sum of every other word in array called SAM; the fields are: Tag, Operation, Operand, Comments

```
Tag     Operation     Operand     Comments
                        .
                        .
                        .
        LAC           (SAM      /LOAD AC WITH
                                /BOTTOM ADDRESS
                                /OF ARRAY
        PAX                     /DEPOSIT INTO INDEX
                                /REGISTER
        LAC           (SAM+N    /LOAD AC WITH
                                /NEXT ADDRESS
                                /AFTER ARRAY
        PAL                     /DEPOSIT INTO LI-
                                /MIT REGISTER
Begin   CLA!CLL                 /CLEAR AC AND
                                /LINK
Loop    TAD           0,X       /FORM SUM
        AXS           2         /ADD TWO TO INDEX
                                /REGISTER AND TEST
                                /IF COMPLETE
        JMP           Loop      /CONTINUE LOOPING
        HALT                    /LOOP COMPLETE
```

!indicates a microprogrammed operation

Auto-Increment Locations - Eight locations (10_8-17_8) of the first 4096-word page act as auto-increment registers. When indirectly addressed, the contents of an auto increment location are incremented by 1 and then taken as the effective address of an instruction. When directly addressed, these locations act like all other memory locations.

The auto-increment locations are used to loop through sequential data arrays. The "increment and skip if zero" (ISZ) instruction is used to test for loop completion. The following example illustrates their use:

```
        Form sum of Am + Bm and store in Cm
                .
                .
                .
Begin   CLA!CLL       /CLEAR AC AND LINK
Loop    LAC* 10       /GET ADDEND
        TAD* 11       /FORM SUM
        DAC* 12       /STORE SUM
```

*indicates indirect addressing

```
        ISZ Count     /TEST FOR COMPLETION
        JMP Loop      /CONTINUE LOOPING
        HALT          /LOOP COMPLETE
Count   -N            /NUMBER OF ITERATIONS,
                      /TWO'S COMPLEMENT
10      L(A)-1        /FIRST LOCATION OF ARRAY
                      /A-1
11      L(B)-1        /FIRST LOCATION OF ARRAY
                      /B-1
12      L(C)-1        /FIRST LOCATION OF ARRAY
                      /C-1
```

BANK MODE ADDRESSING

The PDP-15 can be placed into another mode of addressing called bank mode addressing. In this mode, 8192 words of memory can be directly addressed by memory reference instructions. This is done by allowing the fifth bit of the instruction to refer to a memory bank rather than to the index register. In this mode, all indexing operations including the use of the limit register for register-to-register compares are eliminated in favor of bank addressing. Indirect addressing can be used.

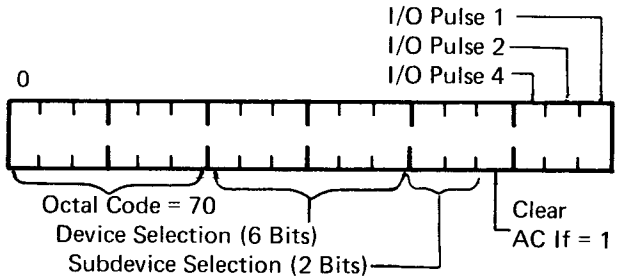
The program counter is incremented modulo 8192. That is, the low-order 13 bits point at the address within the bank specified by the high-order bits. When the program counter is incremented, only the 13 low-order bits function as a counter. There is no carry into the high-order bits. To change banks, a jump indirect instruction is normally used.

An "enable bank addressing" (EBA) instruction places the Central Processor in bank address mode, and a "disable bank addressing" (DBA) instruction places the Central Processor back into the page address mode where indexing can be used.

NONMEMORY REFERENCE INSTRUCTIONS

Input/Output Instruction

IOT instructions are microcoded to effect responses for a particular device. The microcoding includes issuing a unique device selection code and appropriate processor-generated input/output pulses (IOP) to initiate a specific operation. For an "out" transfer, the program reads a data word from memory into the AC. A subsequent IOT instruction places data on the bus, selects the device, and transfers the data to the device. For an "in" transfer, the process is reversed. An IOT instruction selects the device and transfers data into the AC. A subsequent instruction in the program transfers the word from the AC to memory.

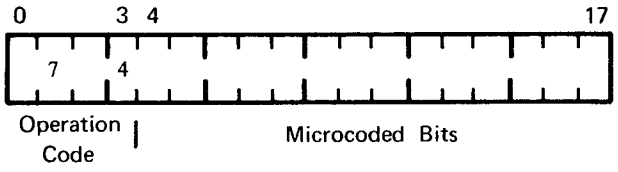


IOT instructions are also used to initialize the single- and multi-cycle channels and to transfer the word count and current address to the single-cycle device controllers. In addition, they are used to test or clear device flags, select modes of device operation and to control a number of processor operations. Within a single IOT instruction, up to 64 unique device-selection codes are available and an additional two bits form up to four subdevice commands. Three microprogrammed pulses (IOP) are also provided to test, initiate transfer, etc. (See the IOT instruction for complete details.)

OPERATE INSTRUCTIONS

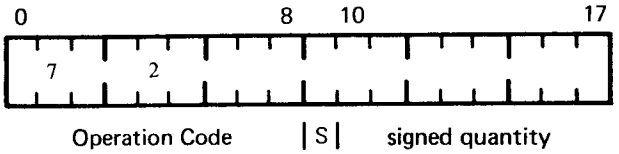
Microcoded Operate Instruction

Microcoded operate instructions (operate code of 74_8) are used to sense and/or alter the contents of the AC and Link. Typical functions are: conditional or unconditional skips and complementing, setting, clearing or rotating the contents of the two registers jointly or independently. A HLT instruction is included. Operates are



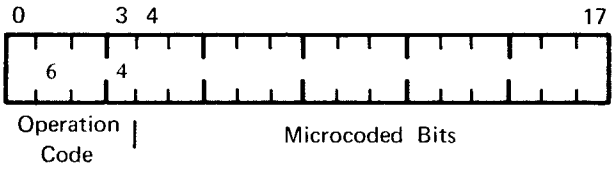
fetched and executed in one machine cycle; the actions are specified by the microprogramming of the instruction code. Each of the 13 bits can effect a unique response; hence they are "microinstructions" to the computer. The important feature of the operate class is its microprogramming capability which allows two or three microinstructions to be combined to form one instruction word and, therefore, to be executed during one cycle. Those microinstructions that logically conflict and occur in the same event time should not be microprogrammed.

Index Operates



Index operate instructions (operate code of 72_8) are used to clear, load and compare the index and limit registers.

EAE Instructions

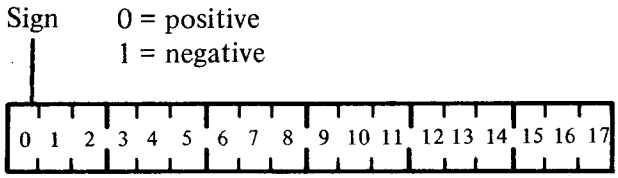


EAE, identified by an operation code of 64_8, performs high-speed data manipulation and multiply divide operations as specified by microprogramming of individual instructions. The microinstructions have capabilities for register set-up, data shift, normalize, multiply, and divide, respectively. EAE is an option on the PDP-15/10 System. For a complete description see the EAE instructions.

DATA WORDS

Memory reference instructions deal with data used for arithmetic and logical operations and are also used for address modifications. The following describes the data formats used in the arithmetic and logical operations.

There are two types of arithmetic instructions in the PDP-15 - 2's complement and 1's complement. Floating point operations are provided in subroutines supplied with the system monitors.



Single-Precision Data

Single-Precision Data

Up to 18 bits of data may be contained in a single precision PDP-15 word. Normally, 2's complement arithmetic is used, because convention has adapted only one representation of 0 in 2's complement notation, namely +0; 1's complement notation has both a +0 and a -0 that can cause ambiguity.

In both complement notations (1's and 2's), the sign indicator (bit 0) is 0 for positive quantities and 1 for negative quantities. The 1's complement of a quantity is equivalent to the logical complement of its magnitude and sign; i.e., all binary 1s are replaced by 0s and all binary 0s are replaced by 1s. The 2's complement of the quantity is equivalent to its 1's complement plus the addition of 1 to the lowest order, or least significant, bit. Positive quantities in either notation have identical representations. For example: $+15_{10}$ is represented in a PDP-15 data word as

```
 s
000 000 000 000 001 111
```

in either 1's or 2's complement notation. The 1's complement of -15_{10} is represented by

```
s
111  111  111  111  110  000
```

The 2's complement of -15_{10} appears as

```
s
111  111  111  111  110  001
```

A typical PDP-15 instruction sequence for forming the 2's complement of any number is:

```
LAC Y
TCMA
DAC Y
```

The TAD (2's complement add) instruction must be used rather than the ADD (1's complement add) instruction as ADD permits an end-around carry into the low-order bit.

Magnitudes of Data Words - For 2's complement signed notation, the permissable magnitude of any quantity, X, is in the range of:

$$-2^{n-1} \leqslant X \leqslant 2^{n-1}-1$$

where n is again the number of bits allocated to the storage of data. A single-precision data word has the range:

$$-2^{17} \leqslant X \leqslant 2^{17}-1$$

or $-131\ 072_{10} \leqslant X \leqslant +\ 131\ 071_{10}$

The position of the decimal point is implied in the above ranges.

For 1's complement signed and sign-and-magnitude notations, the permissable magnitude of any quantity, X, is in the range of:

$$-(2^{n-1}-1) \leqslant X \leqslant 2^{n-1}-1$$

where n is the number of bits allocated to the storage of data in a data word. For a single-precision data word (sign bit and 17 data bits), this relationship becomes:

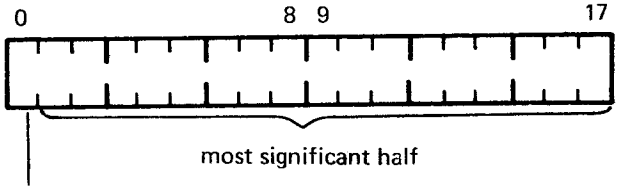
$$-(2^{17}-1) \leqslant X \leqslant 2^{17}-1$$

$$-131\ 071_{10} \leqslant X \leqslant +\ 131\ 071_{10}$$

Double Precision Data

A signed double-precision data word consists of two computer words for a total of 36 bits (Figure 6-4). The first word contains the sign bit and the 17 most-significant bits; the second word contains the 18 least-significant bits. The words are stored in consecutively addressed core memory locations for ease of programming.

1st Word



Sign 0 = positive
1 = negative

2nd Word

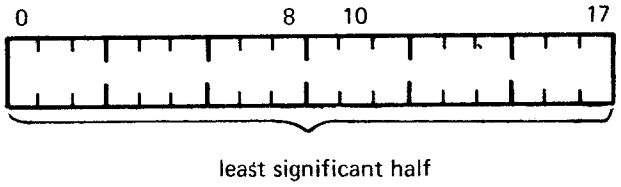


Figure 6-4. Double-Precision Data Word Formats

The magnitude of double precision in 2's complement is

$$-2^{35} \leqslant X \leqslant 2^{35}-1$$

$$-34{,}359{,}738{,}368 \leqslant X \leqslant 34{,}359{,}738{,}367$$

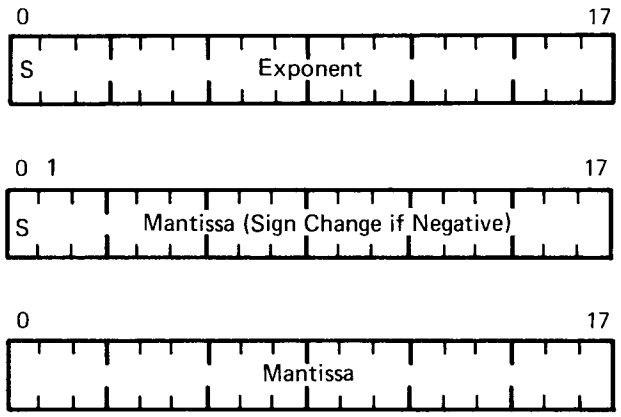
BASIC SOFTWARE FLOATING-POINT FORMATS

Floating-point representation of a binary number consists of two parts: the exponent and the mantissa. The mantissa is a fraction with the binary point assumed to be positioned between the sign bit and the most-significant data bit. The mantissa is always stored in a normalized state, i.e., leading 0s are eliminated from the binary representation so that the high-order bit is always a 1. The exponent, as stored, represents the power of 2 by which the mantissa is multiplied to obtain the number's value for use in computation.

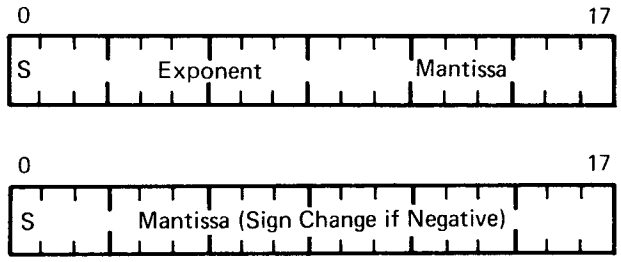
The PDP-15 floating-point software system offers two modes for storage of floating-point numbers: three-word mode and two-word mode.

The three-word mode requires three memory locations for storage of a floating-point binary number (Figure 6-5a). The exponent, a signed 17-bit integer in 2's complement notation, occupies the first word, or memory location. The mantissa, a 35-bit quantity in sign and magnitude notation, is stored in the second and third words. The sign of the mantissa is stored in the high-order bit of the second word. The range is $\pm 10^{39000}$ accurate to 9 decimal digits.

The two-word mode requires two memory locations for storage of a floating-point binary number (Figure 6-5b). The exponent, an 8-bit integer in 2's complement notation, and its sign occupy the 9 high-order bits of the first word. The mantissa, a 26-bit quantity in sign and magnitude notation, is stored in the 9 low-order bits of the first word and in the 17 low-order bits of the second word. The sign of the mantissa is stored in the high-order bit of the second word. The range is $\pm 10^{60}$ with an accuracy to 6 decimal digits.



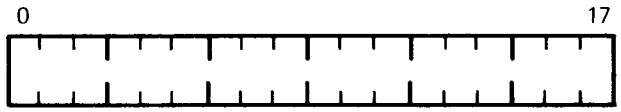
a. Three-Word Mode



b. Two-Word Mode

Figure 6-5. Floating Point Formats

BOOLEAN REPRESENTATION



Boolean operations use unsigned quantities.

Full 18-bit words can be ANDed or exclusive ORed with each other.

CHAPTER 7
INSTRUCTION REPERTOIRE

This chapter describes the instruction set of the basic PDP-15 by function, and describes the use and action of each operation. The format is as follows:

Mnemonic — Three to six alphabet characters which represent the operation code in the MACRO-15 Assembler.

Operation Name — A description of the instruction.

Octal Code — The machine language code in octal notation. The symbols listed below will be used where applicable.

A	The bits of the Address Field
E	The E Field
I	Indirect Addressing
X	Index Addressing

Execute Time — The times shown herein are worst case times for a system without memory parity.

INSTRUCTION GROUPS

Memory Reference Instructions

The specific operations of the PDP-15 instruction repertoire, are categorized into the following groups:

Transfer Instructions

DAC	Deposit Accumulator
LAC	Load Accumulator
DZM	Deposit Zero in Memory

Arithmetic Instructions

ADD	Add, 1's Complement
TAD	Add, 2's Complement
ISZ	Increment and Skip if Zero

Logical Instructions

XOR	Exlusive OR
AND	AND (Logical Product)
SAD	Skip if Accumulator different from Memory

Jump and Skip Instructions

CAL	Call Subroutine
JMP	Unconditional Jump
JMS	Jump to Subroutines

Control Instructions

XCT	Execute

Operate Instructions

Rotate Instructions

RAR	Rotate Accumulator and Link 1 Right
RTR	Rotate Accumulator and Link 2 Right
RAL	Rotate Accumulator and Link 1 Left
RTL	Rotate Accumulator and Link 2 Left

Control Instructions

OPR or NOP	No Operation
HLT	Halt
CLL	Clear Link
CML	Complement Link
STL or CCL	Set Link
GLK	Get the Link
CLA	Clear Accumulator
CLC	Clear and Complement Accumulator
OAS	OR Console Accumulator Switches to Accumulator
IAC	Increment the Accumulator
SWHA	Swap Halves of the Accumulator
LAW	Load Accumulator with this Instruction
CMA	1's complement the Accumulator
TCA	2's complement the Accumulator
SKP	Skip Unconditionally

Skip on Register Condition Instructions

SPA	Skip if Accumulator is Positive
SMA	Skip if Accumulator is Negative
SNA	Skip if Accumulator Not Zero
SZA	Skip if Accumulator is Zero
SNL,SML	Skip if Link Not Zero
SZL,SPL	Skip if Link Equals Zero

Microcoded Instructions

LAS=CLA!OAS	Load Accumulator from Console
LAS!CMA	Load Accumulator from 1's Complement of Console
CLC=CLA!CMA	Set Accumulator to all 2's
STL=CLL!CML	Set the Link to 1
STL!CLC	Set Link and Accumulator to all 1's
RCL=CLL!RAL	Clear Link and Rotate Left
RCR=CLL!RAR	Clear Link and Rotate Right

Index and Limit Register Instructions

Register Transfer Instructions

PAX	Place Accumulator in Index Register
PAL	Place Accumulator in Limit Register
PXA	Place Index Register in Accumulator
PXL	Place Index Register in Limit Register
PLA	Place Limit Register In Accumulator
PLX	Place Limit Register in Index Register

Register Control Instructions

AAS n	Add n to Accumulator and skip if result is equal to or greater than Limit Register
AXS n	Add n to Index Register and skip if result is equal to or greater than the limit Register

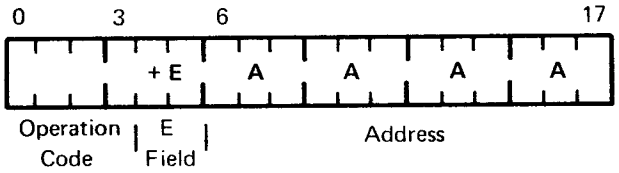
AXR n	Add n to the Index Register
AAC n	Add n to Accumulator
CLX	Clear the Index Register
CLAC	Clear the Accumulator
CLLR	Clear the Limit Register

Input/Output Instructions

IORS	Read Flags
CAF	Clear All Flags
IOF	Turn Interrupt OFF
ION	Turn Interrupt ON

TRANSFER INSTRUCTIONS

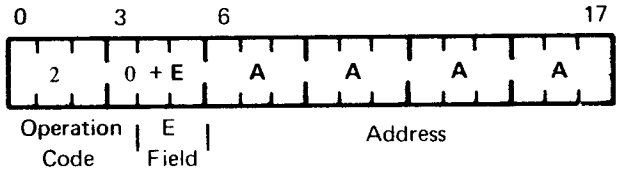
DAC **Deposit Accumulator**



Execute Time: 1.6 μs

The content of the Accumulator replaces the content of the memory location specified by the effective address. The accumulator remains unchanged.

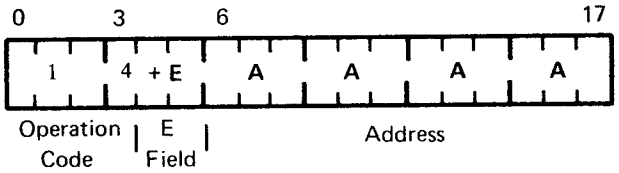
LAC **Load the Accumulator**



Execute Time: 1.6 μs

The content of the memory location specified by the effective address replaces the contents of the accumulator. The content of the memory location is unchanged.

DZM **Deposit Zero in Memory**

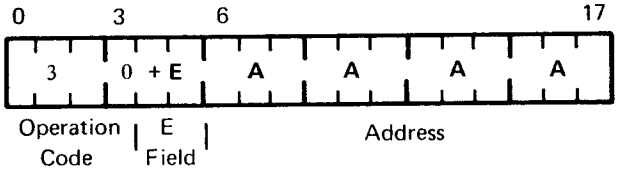


Execute Time: 1.6 μs

The content of the memory location specified by the effective address is zeroed. All other registers remain unchanged.

ARITHMETIC INSTRUCTIONS

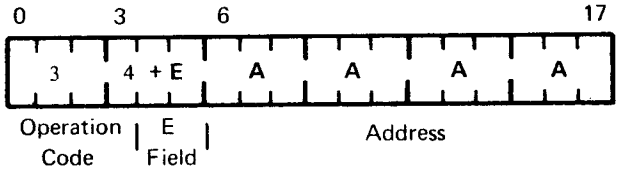
ADD **ADD, 1's Complement**



Execute Time: 1.6 μs

The content of the effective address and the content of the accumulator are added in 1's complement arithmetic. The results are put into the accumulator.

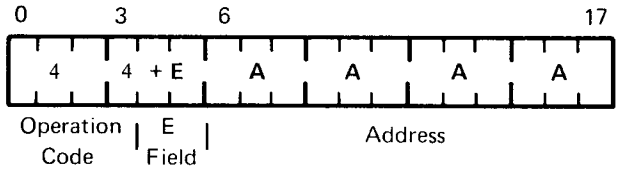
TAD **ADD, 2's Complement**



Execute Time: 1.6 μs

The content of the effective address and the content of the accumulator are added in 2's complement arithmetic. The results are put into the accumulator.

ISZ **Increment and Skip if Zero**

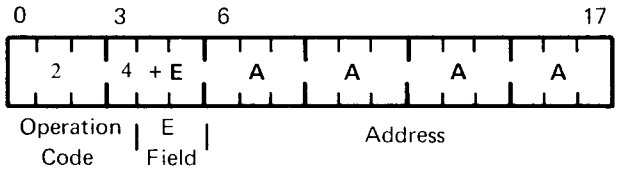


Execute Time: 2.4 μs

The content of the memory location specified by the effective address is incremented by 1. Two's complement arithmetic is used for the algebraic sum. If the result in memory is equal to 0, the next instruction is skipped. If the result is not 0, the next sequential instruction is executed. The content of the accumulator is unchanged.

LOGICAL INSTRUCTIONS

XOR **Exclusive OR (Half Add)**



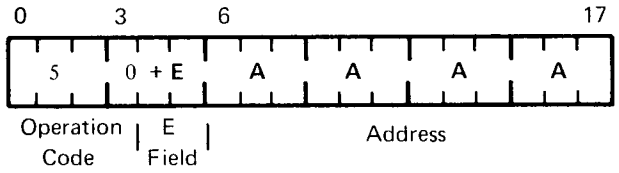
Execute Time: 1.6 μs

The Exclusive ORed bits of the accumulator and the content of the memory location specified by the effective address, C(EA), replace the content of the accumulator.

Programming Note - The XOR instruction causes the operand to complement its original content only in those bits which have 1s in the accumulator mask. The truth table for the XOR instruction is -

(AC)		C(EA)		RESULT
0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	0

AND **AND (Logical Product)**



Execute Time: 1.6 μs

The logical product of the bits of the accumulator and the content of the memory location specified by the effective address replaces the content of the accumulator.

Programming Note - A logical product is used to select or mask specific portions of an operand. If only a selected portion of an operand is required, it is subjected to a mask placed in the accumulator. The mask is composed of patterns of 0s and 1s.

The AND instruction causes the bits of the operand to appear unchanged in the accumulator only in the area of the original mask of 1s bits. The AND instruction can be considered as "both bits high."

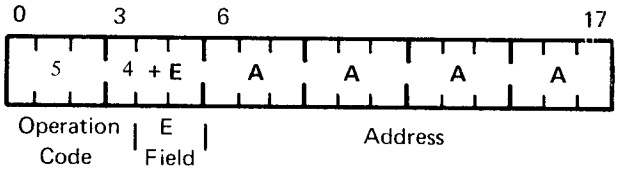
For example, to obtain the displacement address for an instruction (least-significant 12 bits), the following operation would be performed.

Operand Content	010110111011110101
Accumulator Mask	000000111111111111
Accumulator Result	000000111011110101

The truth table for the AND instruction is as follows:

(AC)		C(EA)		RESULT
0	x	0	=	0
0	x	1	=	0
1	x	0	=	0
1	x	1	=	1

SAD **Skip if Accumulator Different From Memory**

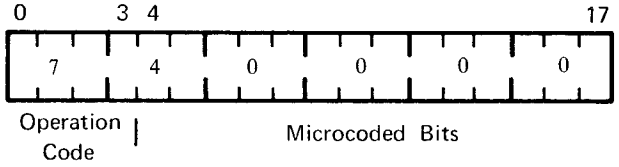


Execute Time: 1.6 μs

The content of the memory location specified by the effective address is compared algebraically with the content of the accumulator. If they are not equal, the program counter is incremented by 1, and thus the next instruction is skipped. Neither the contents of the accumulator nor the contents of the specified memory location are altered.

ROTATE INSTRUCTIONS

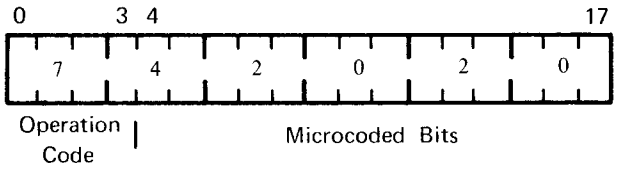
RAR **Rotate Accumulator and Link One Right**



Execute Time: 800 ns

The contents of the accumulator and Link are rotated one bit to the right. The content of the Link moves into accumulator bit 0, and accumulator bit 17 moves into the Link.

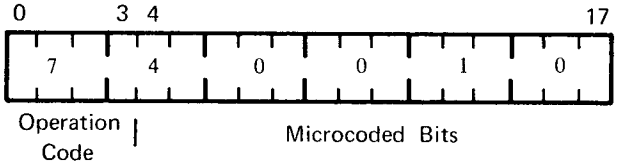
RTR **Rotate Accumulator and Link Two Right**



Execute Time: 800 ns

The contents of the accumulator and Link are rotated two bits to the right. The content of the Link moves into accumulator bit 1, accumulator bit 17 moves into bit 0 and bit 16 moves into the Link.

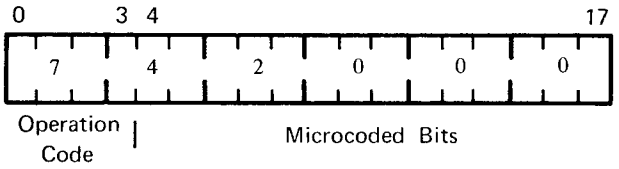
RAL **Rotate Accumulator and Link One Left**



Execute Time: 800 ns

The contents of the accumulator and Link are rotated one bit to the left. Accumulator bit 0 moves into the Link, and the Link moves to accumulator bit 17.

RTL **Rotate Accumulator and Link Two Left**

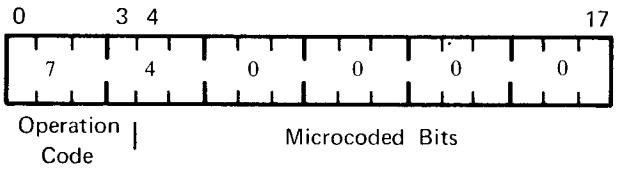


Execute Time: 800 ns

The contents of the accumulator and Link are rotated two bits to the left. The Link moves to accumulator bit 16, accumulator bit 0 moves to bit 17 and accumulator bit 1 moves to the Link.

CONTROL INSTRUCTIONS

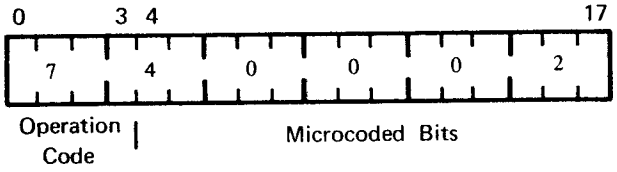
OPR or NOP **No Operation**



Execute Time: 800 ns

The OPR or NOP instruction causes a delay of one cycle. The program counter is incremented and processing continues.

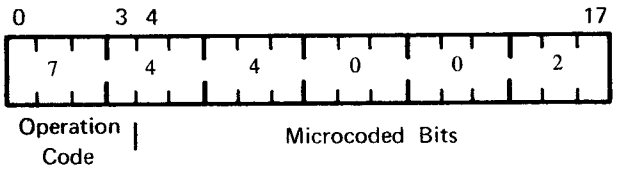
CML **Complement the Link**



Execute Time: 800 ns

The content of the Link is complemented.

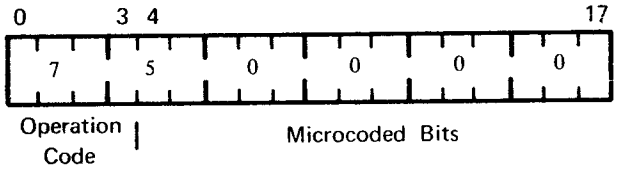
STL or (CCL) **Set the Link (Clear and Complement the Link)**



Execute Time: 800 ns

The content of the Link is set to a one.

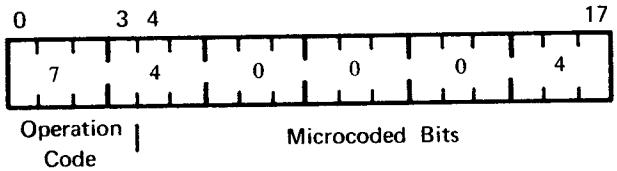
CLA **Clear the Accumulator**



Execute Time: 800 ns

The content of the AC is zeroed.

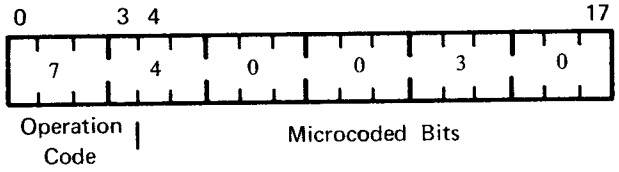
OAS **OR Console Accumulator Switches to the Accumulator**



Execute Time: 800ns

The contents of the console accumulator switches are inclusive ORed with the contents of the accumulator, and the results are put into the accumulator.

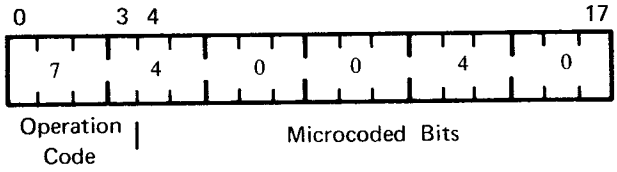
IAC **Increment the Accumulator**



Execute Time: 800ns

The content of the accumulator is incremented by 1 in 2's complement. A carry-out complements the Link.

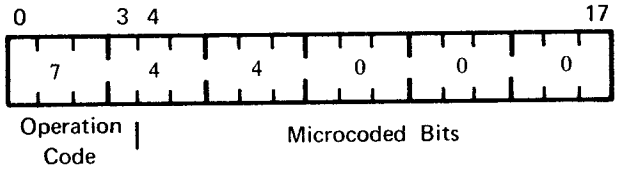
HLT **Halt**



Execute Time: 800ns

The computer is stopped and must be restarted manually by depressing the console CONTINUE or RESTART buttons. CONTINUE begins execution at the instruction immediately following the Halt instruction and RESTART loads the contents of the switch register into the location counter and begins execution at that address.

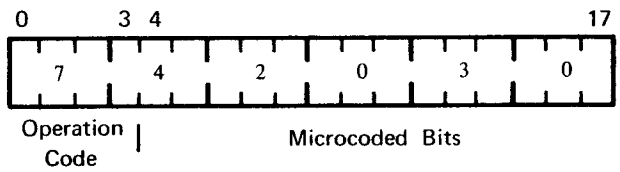
CLL **Clear Link**



Execute Time: 800ns

The content of the Link is set to zero.

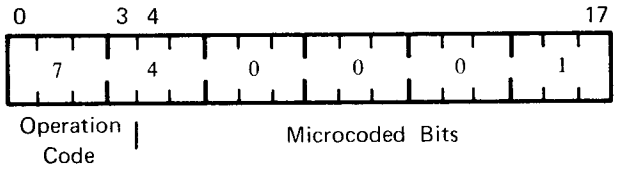
SWHA **Swap Halves of the Accumulator**



Execute Time: 800ns

The high-order 9 bits of the accumulator (0-8) are exchanged with the low-order bits (9-17). The content of the Link is unchanged.

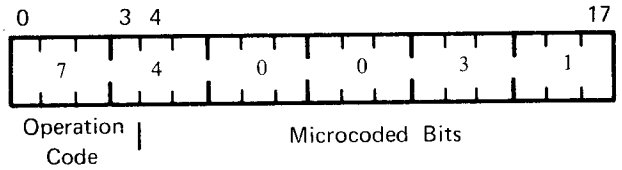
CMA — **One's Complement the Accumulator**



Execute Time: 800 ns

The content of the accumulator is replaced by its 1's complement; i.e., each bit in the accumulator is inverted.

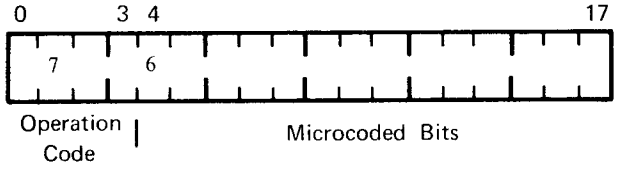
TCA — **Two's Complement the Accumulator**



Execute Time: 800 ns

The content of the accumulator is replaced by its 2's complement.

LAW — **Load Accumulator with This Instruction**

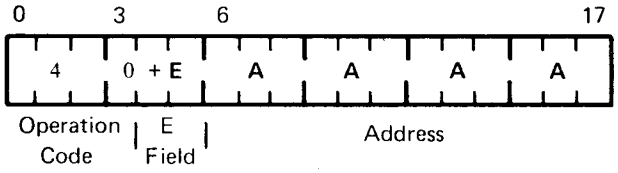


Execute Time: 800ns

This instruction loads itself into the accumulator $0 \leqslant N \leqslant 17777$.

Programming Note - This instruction is used for loading alphanumeric character codes into the accumulator for transfer to peripherals, to initialize word count or current address locations and to preset the real time count. LAW should not be used for address formulation since program control by this instruction is only for the lower 8K of memory.

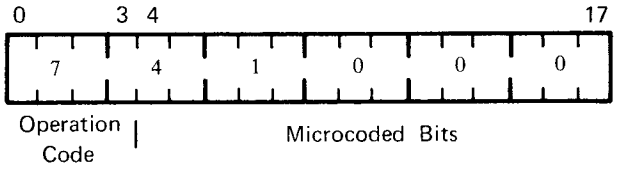
XCT — **Execute**



Execute Time: 800 ns + Instruction Time
[e.g. XCT (LAC A) = 0.8 + 1.6 = 2.4μs]

The effective address is calculated and the instruction stored there is executed. The program counter remains unaltered unless the instruction of the effective address changes it (if it is JMP, JMS, CAL or SKIP).

SKIP — **Skip Unconditionally**

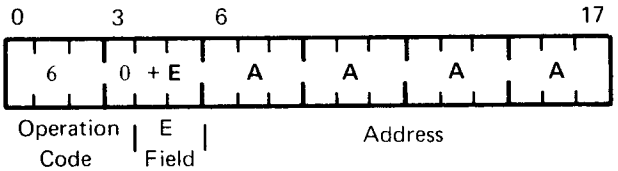


Execute Time: 800ns

The next sequential instruction is unconditionally skipped.

JUMP INSTRUCTIONS

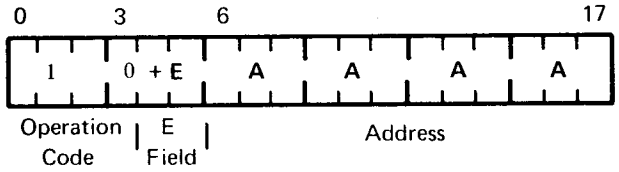
JMP — **Unconditional Jump**



Execute Time: 800ns

The effective address is computed and replaces the current content of the program counter register.

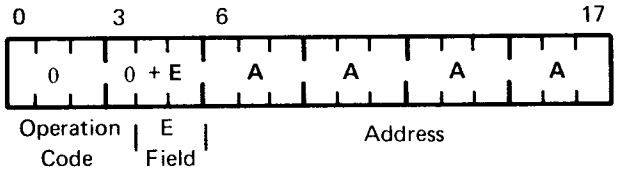
JMS **Jump to Subroutine**



Execute Time: 1.6 μs

The content of the program counter, the Link, and the status of the memory protect mode replace the content of the memory location specified by the effective address. The content of the program counter is stored in bits 3 through 17, memory protect mode in bit 2 and Link in bit 0. The effective address plus one replaces the content of the program counter.

CAL **Call Monitor**

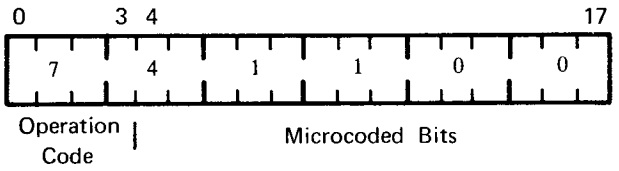


Execute Time: 1.6 μs

CAL specifies a JMS 20 to page 0 regardless of the content of the address field. This instruction is used to call the monitor while the system is in user mode. The address field stores information for the monitor from the user. CAL automatically sets the API to level 4.

If an index bit is set in a CAL instruction, it is ignored; i.e., no indexing is done. An indirect bit in the CAL instruction, however, will cause a JMS I 20.

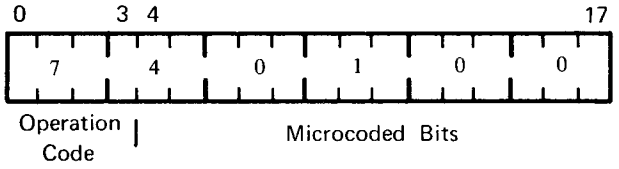
SPA **Skip if Accumulator is Positive**



Execute Time: 800ns

The sign (bit 0) of the content of the accumulator is tested and, if positive (0), the next instruction is skipped. If the sign is negative (1), the next instruction is executed. The content of the accumulator remains unchanged.

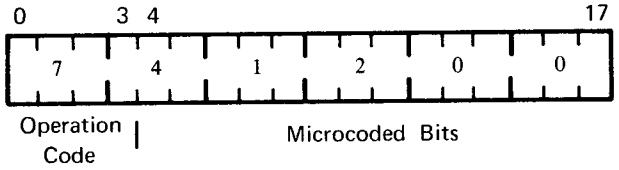
SMA **Skip if Accumulator is Negative**



Execute Time: 800 ns

The sign (bit 0) of the content of the accumulator is tested and, if negative (1), the instruction is skipped. If the sign is positive, the next sequential instruction is executed. The content of the accumulator is not altered.

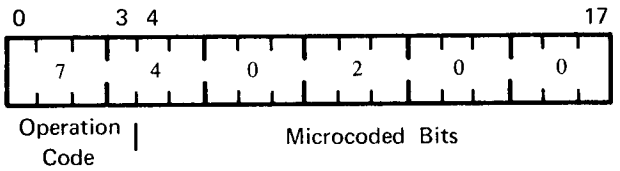
SNA **Skip if Accumulator is Not Zero**



Execute Time: 800 ns

The content of the accumulator is tested. If any bits are 1s, the next instruction is skipped. If all bits are 0s, the next instruction is executed. The content of the accumulator remains unchanged.

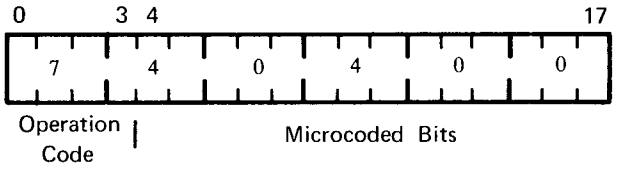
SZA **Skip if Accumulator is Zero**



Execute Time: 800 ns

The content of the accumulator is tested and if all bits are 0, the next instruction is skipped. If any bits are a 1 the next instruction is executed. The contents of the accumulator remain unchanged.

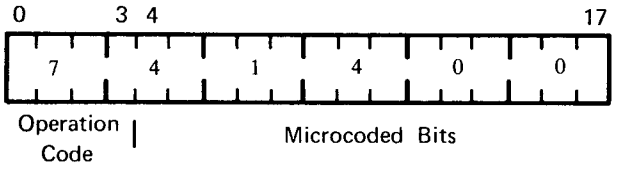
SNL or SML **Skip if Link Not Zero**
Skip on Minus Link



Execute Time: 800 ns

The content of the Link is tested. If it is not 0, then the next instruction is skipped. If it is a 0, the next instruction is executed. The content of the Link remains unchanged.

SZL or SPL **Skip if Link Equal Zero**
Skip on Positive Link



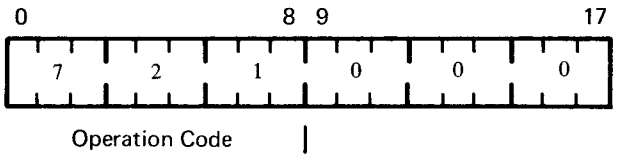
Execute Time: 800 ns

The content of the Link is tested. If it is a 0, then the next instruction is skipped. If it is a 1, the next instruction is executed. The content of the Link remains unchanged.

INDEX AND LIMIT REGISTER INSTRUCTIONS

Register Transfer Instructions

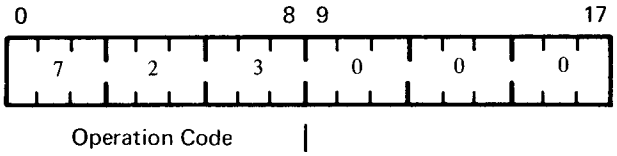
PAX **Place Accumulator in Index Register**



Execute Time: 1.6 μs

The content of the index register replaces the content of the accumulator. The content of the index register remains unchanged.

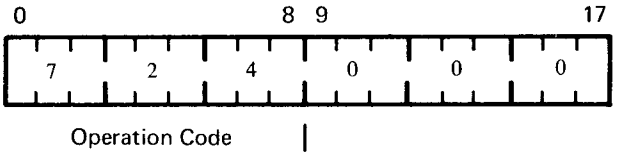
PAL **Place Accumulator in Limit Register**



Execute Time: 1.6 μs

The content of the limit register is replaced by the content of the accumulator. The content of the accumulator remains unchanged.

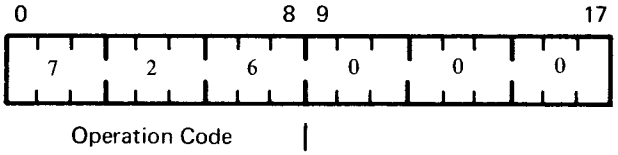
PXA **Place Index Register in Accumulator**



Execute Time: 1.6 μs

The content of the index register replaces the content of the accumulator. The content of the index register remains unchanged.

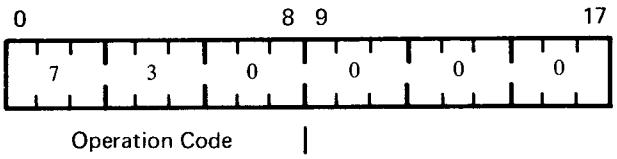
PXL **Place Index Register in Limit Register**



Execute Time: 1.6 μs

The content of the index register replaces the content of the limit register. The content of the index register remains unchanged.

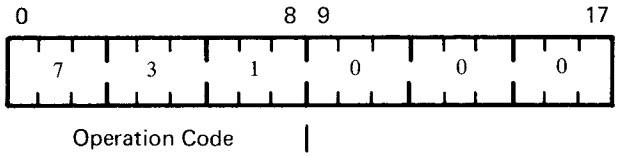
PLA **Place Limit Register in Accumulator**



Execute Time: 1.6 μs

The content of the limit register replaces the content of the accumulator. The content of the limit register remains unchanged.

PLX **Place Limit Register in Index Register**

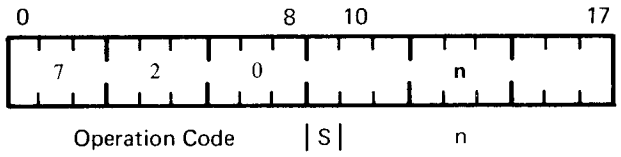


Execute Time: 1.6 μs

The content of the limit register replaces the content of the index register. The content of the limit register remains unchanged.

REGISTER CONTROL INSTRUCTIONS

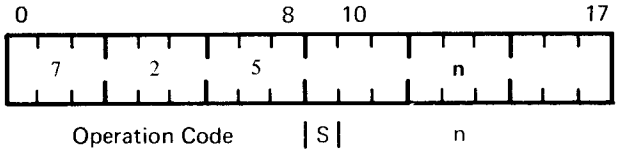
AAS n **Add n to Accumulator and Skip if Result Equal to or Greater Than Limit Register**



Execute Time: 1.6 μs

n, a signed 9-bit (8 bits plus sign) 2's complement integer is added to the content of the accumulator, and the results are placed in the accumulator. If the sum is greater than or equal to the content of the limit register, then the program counter is incremented by 1 and thus the next instruction is skipped.

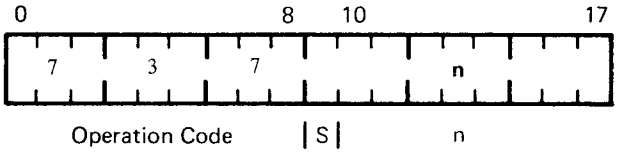
AXS n **Add n to Index Register and Skip if the Result Equal to or Greater than the Limit Register**



Execute Time: 1.6 μs

n, a signed 9-bit (8 bits plus sign) 2's complement integer is added to the content of the index register, and the results are placed in the index register. If the sum is greater than or equal to the content of the limit register, then the program counter is incremented by 1 and thus the next instruction is skipped.

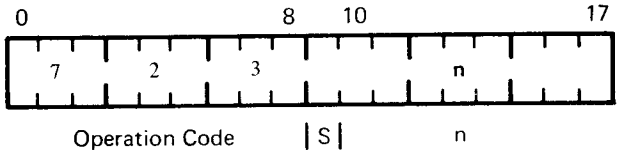
AXR n **Add to Index Register**



Execute Time: 1.6 μs

n, a signed 9-bit (8 bits plus sign) 2's complement integer is added to the content of the index register, and the result is placed in the index register.

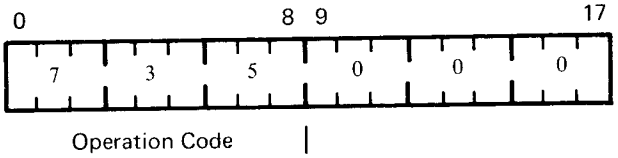
AAC n **Add n to Accumulator**



Execute Time: 1.6 μs

n, a signed 9-bit (8 bits plus sign) 2's complement binary number, is added to the content of the accumulator, and the result is placed into the accumulator.

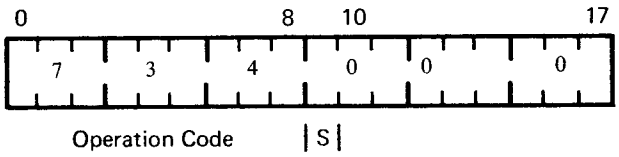
CLX **Clear the Index Register**



Execute Time: 1.6 μs

The content of the index register is replaced with all 0s. Former content is lost.

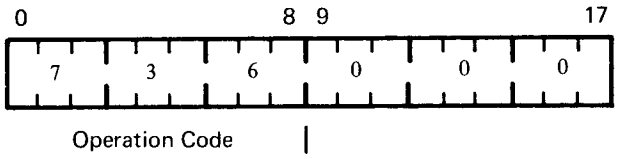
CLAC **Clear the Accumulator**



Execute Time: 1.6 μs

The content of the accumulator is replaced with all 0s. The former content is lost.

CLLR **Clear the Limit Register**



Execute Time: 1.6 μs

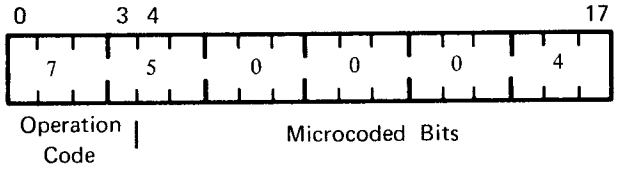
The content of the limit register is replaced with all 0s. The former content is lost.

MICROCODING

Some of the preceding instructions, which do not reference memory, can be combined by inclusive ORing their codes. Figure 7-1 gives the set of all possible pairs of these instructions, and indicates which can be meaningfully combined to form higher order sets. From this figure, sets of triplets can also be found, to form even more complex instructions. A summary of the more useful combinations follows.

Microcoded Instructions

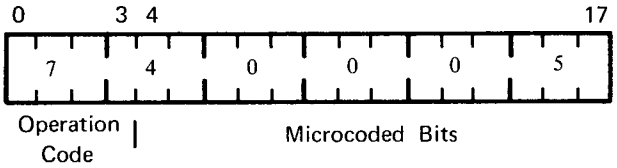
LAS = CLA!OAS **Load Accumulator from Console**



Execute Time: 800 ns

The content of the console switches replaces the content of the accumulator.

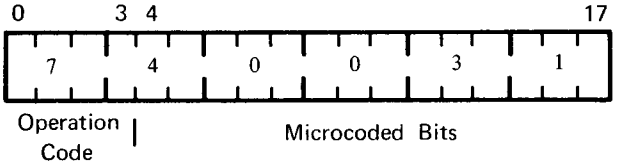
LAS ! CMA **Load Accumulator From Console in 1's Complement**



Execute Time: 800 ns

The 1's complement of the content of the console accumulator switches replaces the content of the accumulator.

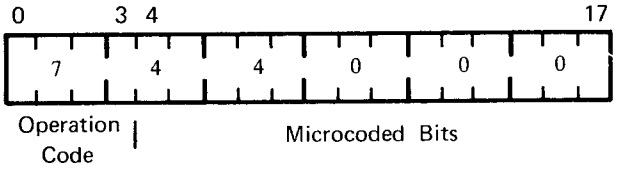
TCA = CMA!IAC **2's Complement the Accumulator**



Execute Time: 800 ns

The content of the accumulator is replaced by its 2's complement.

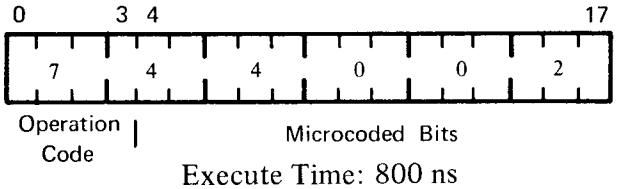
CLC = CLA!CMA **Set the Accumulator to All Ones**



Execute Time: 800 ns

The content of the accumulator is set to all 1's. The Link is unchanged.

STL= CLL!CML **Set the Link to One**



Execute Time: 800 ns

The content of the Link is set to a 1.

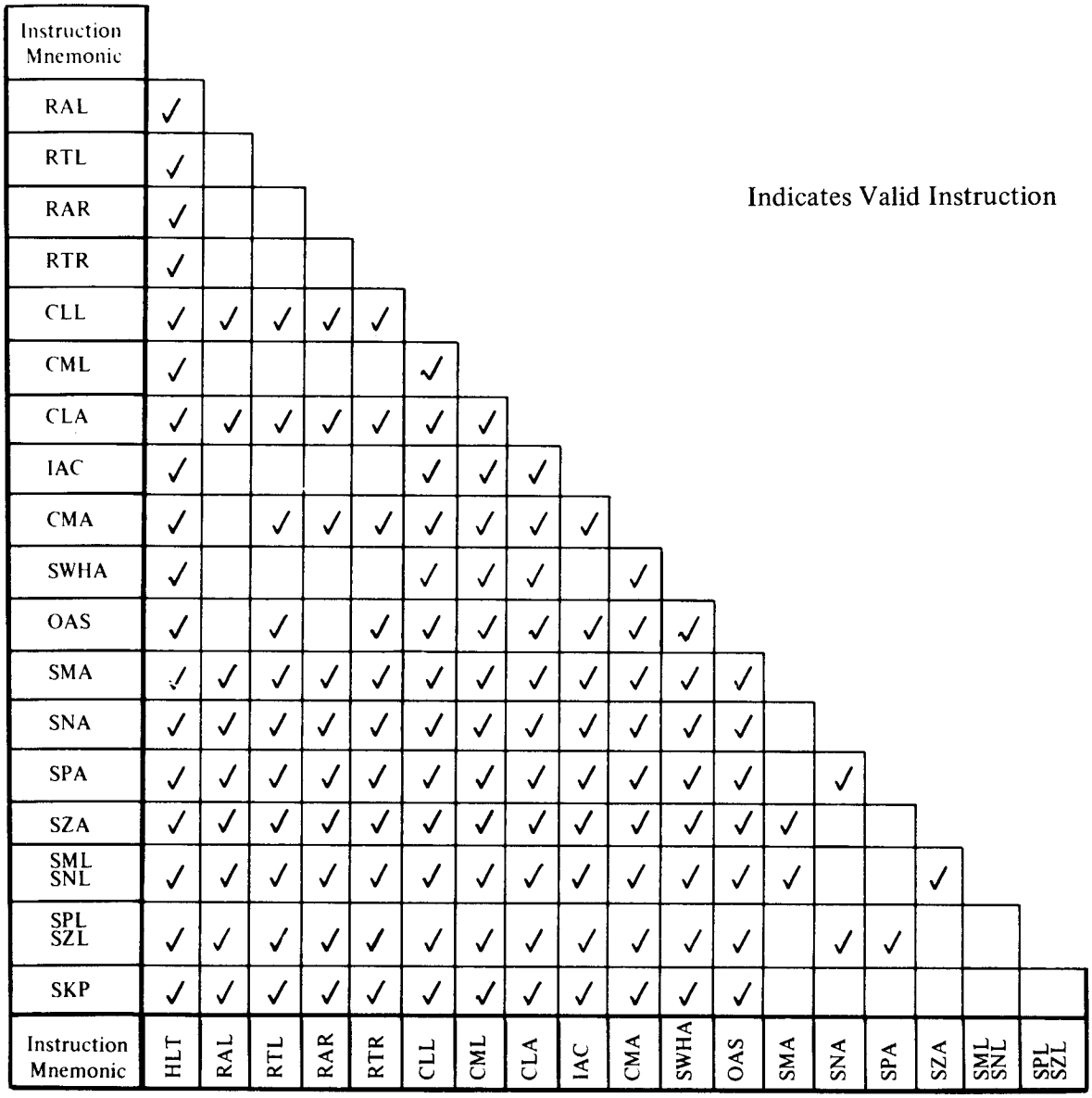
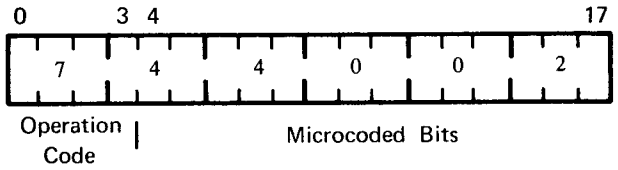


Figure 7-1. Microcoding PDP-15 Control Instructions

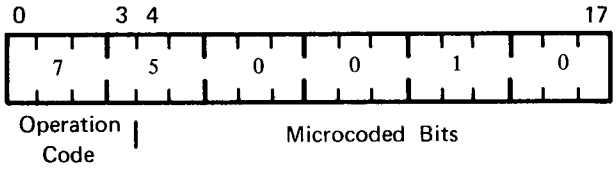
STL! CLC **Set Link and Accumulator to All Ones**



Execute Time: 800 ns

Both the Link and the accumulator are set to all 1's.

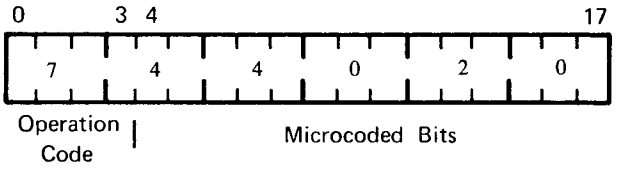
GLK = CLA! RAL **Get the Link**



Execute Time: 800 ns

The accumulator is cleared and the Link shifts into accumulator bit 17.

RCR = Clear Link and Rotate Right
CLL! RAR



Execute Time: 800 ns

The Link is cleared to 0, and then the Link and accumulator are rotated one bit to the right.

INPUT/OUTPUT INSTRUCTION GROUP

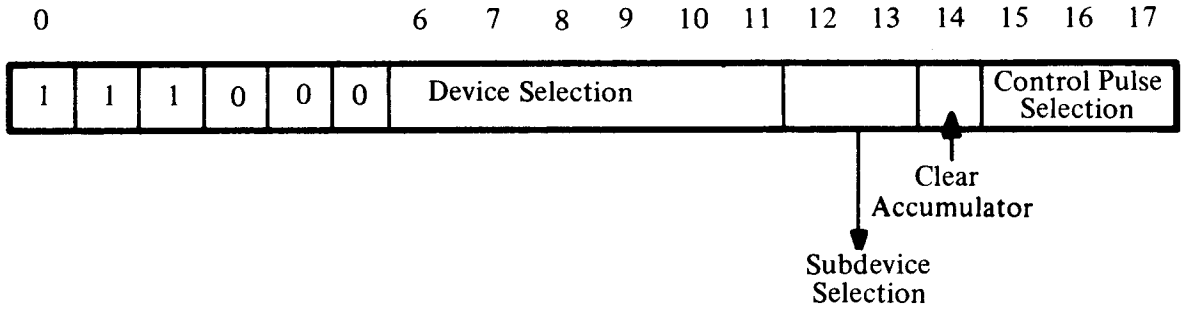
Each internal or peripheral device is assigned a subgroup of the group of instructions called Input/Output Instructions. The format for this group is shown below:

Bit-14 asserted clears the accumulator before a transfer

Bits 15, 16 and 17 select three control pulses IOP4, IOP2, and IOP1, respectively.

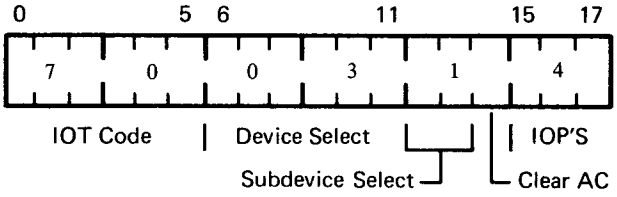
The operate code is 70

The device and subdevice bits select the device's code or address



Details on the use of input/output instructions are given in the PDP-15 User Handbook or the PDP-15 Interface Manual. Those instructions which are used in the basic PDP-15 are:

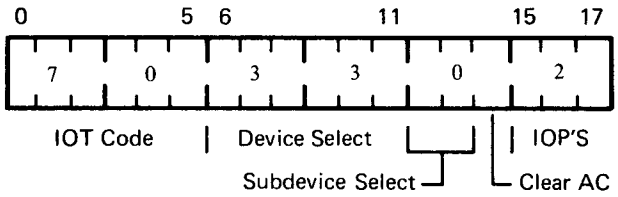
IORS **Read Flags**



Execute Time: 4 μs

The read flags instruction causes the transfer of an 18-bit system status word from the I/O bus to the accumulator. During this instruction, each of the internal and external system devices gates its status register onto preassigned data lines. The I/O Processor transfers these bits to the accumulator. Figure 7-2 shows the word/status/ bit assignment.

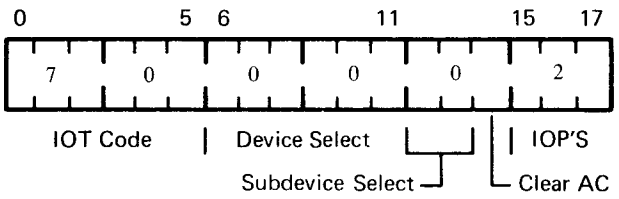
CAF **Clear all Flags**



Execute Time: 4 μs

This instruction gates a pulse to the I/O bus to initialize (clear) all flags of any device that can call for interrupt service. Customer-installed equipment should make use of this pulse to reset flags and registers that must be cleared for system initiation.

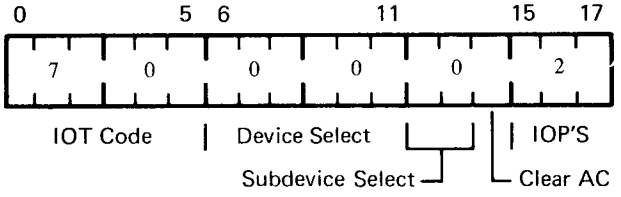
IOF **Turn Interrupt OFF**



Execute Time: 4 μs

This input/output instruction turns off the program interrupt facility of the exchange.

ION **Turn Interrupt ON**



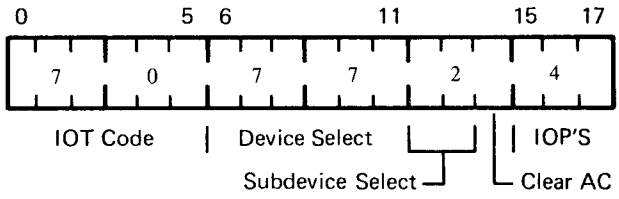
Execute Time: 4 μs

The program interrupt facility is enabled.

NOTE

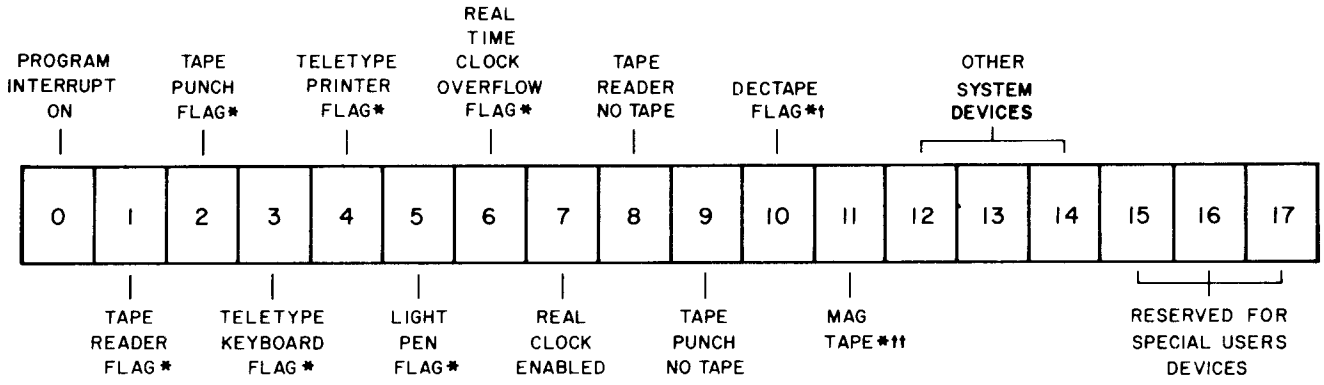
For detailed information on programming with the program interrupt facility, see the PDP-15 User Handbook or the PDP-15 Applications Manual.

EBA **Enable Bank Addressing**



Execute Time: 4 μs

The index register is disabled and the thirteenth bit, normally used to indicate an indexed operation, is gated to the memory address field, permitting direct addressing of 8,192 words of memory.

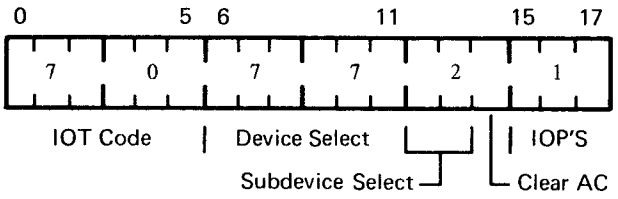


* WILL CAUSE A PROGRAM INTERRUPT
*† INCLUSIVE OR OF TRANSFER COMPLETION AND ERROR FLAGS
*†† INCLUSIVE OR OF MTF AND EF

15-0021

Figure 7-2. IORS Word Status Bit Assignments

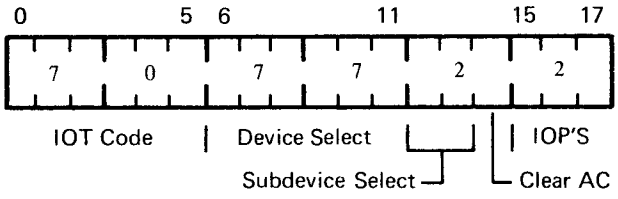
SBA **Skip if in Bank Addressing**



Execute Time: 800 ns

If in bank addressing mode the next instruction is skipped.

DBA **Disable Bank Addressing**



Execute Time: 4 μs

Bank address mode is disabled, and the PDP-15 operates with indexing and addresses 4096 words of memory directly.

CHAPTER 8
INTERNAL OPTIONS INSTRUCTION SET

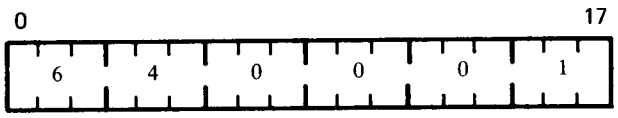
The options available with PDP-15 systems and the instruction sets for each option are described in this chapter.

EAE

The Extended Arithmetic Element, a Central Processor option, facilitates high-speed multiplication, division, shifting, normalizing and register manipulation. Installation of this element as shown in Figure 8-1 adds an 18-bit multiply quotient register, step counter and additional shift logic to the arithmetic unit, shown in Figure 1-4. The instruction set associated with this unit is as follows.

Control Instructions

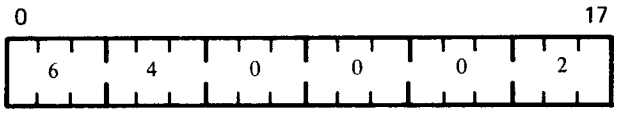
OSC **Inclusive OR The Step Counter With the Accumulator**



Execute Time: 1.75 μs

The content of the step counter is inclusively ORed with accumulator bits 12 to 17, and the result replaces bits 12 to 17 of the accumulator. The contents of accumulator bits 0 to 11, and the step counter are unchanged.

OMQ **Inclusive OR Multiply Quotient Register with the Accumulator**



Execute Time: 1.75 μs

The content of the multiply quotient register is inclusively ORed with the content of the accumulator, and the results replace the former contents of the accumulator. The content of the multiply quotient register is unchanged.

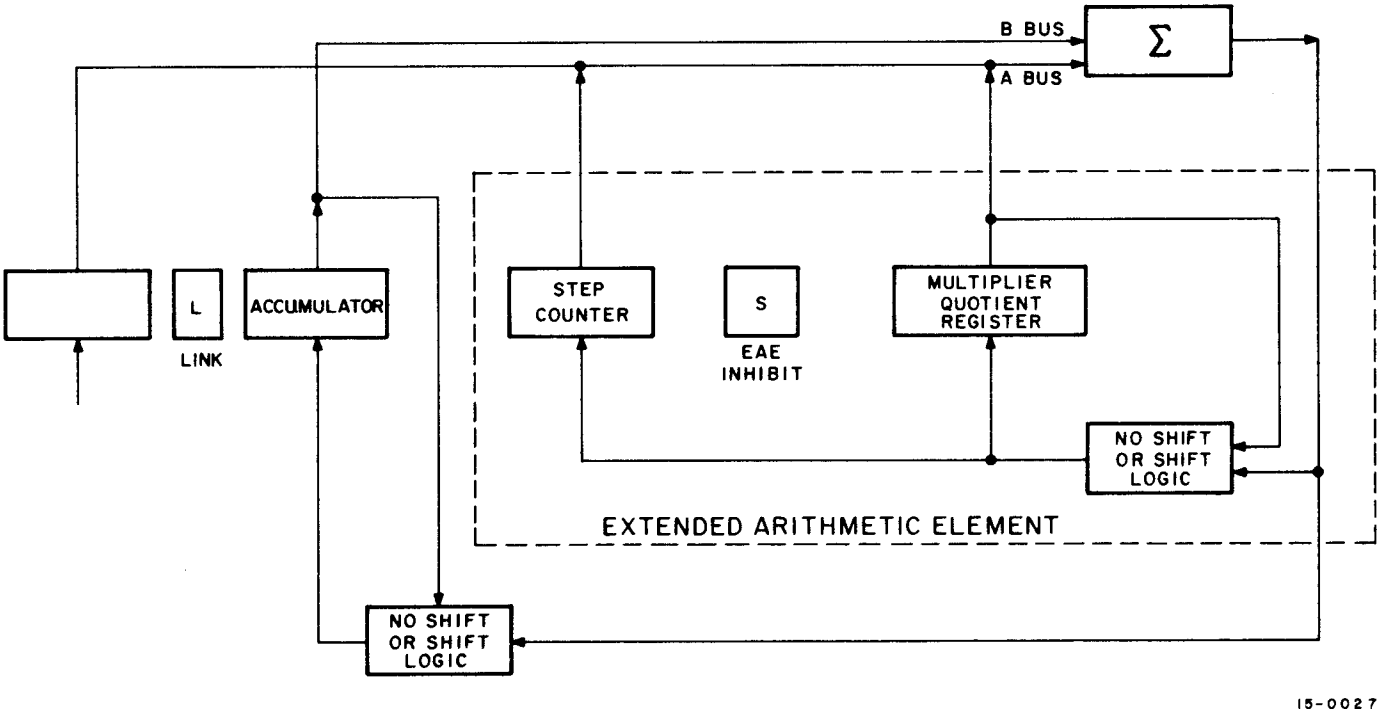
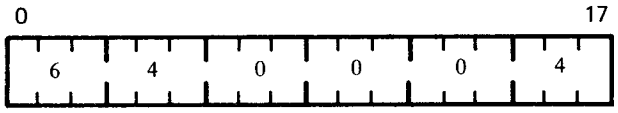


Figure 8-1. Simplified Arithmetic Unit With The Extended Arithmetic Element

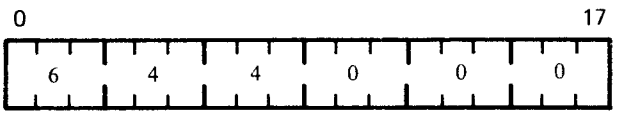
CMQ **Complement the Multiply Quotient Register**



Execute Time: 1.75 μs

The 1's complement (logical inversion) of the multiply quotient register replaces the former content of the multiply quotient register.

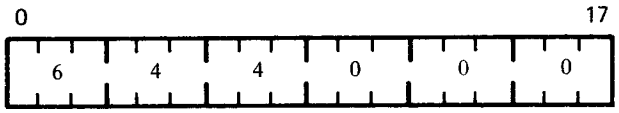
ABS **Replace the Contents of the Accumulator with the Absolute Value**



Execute Time: 1.75 μs

If the content of the accumulator is negative - that is, if bit 0 is a 1, then the content is 1's complemented (logical inversion).

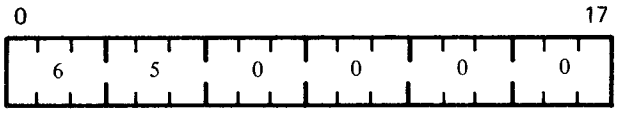
OAC **Inclusive OR Accumulator and Multiply Quotient Register**



Execute Time: 1.75 μs

The contents of the accumulator and the multiply quotient register are inclusive ORed, and the result is placed into the multiply quotient register. The content of the accumulator is unchanged.

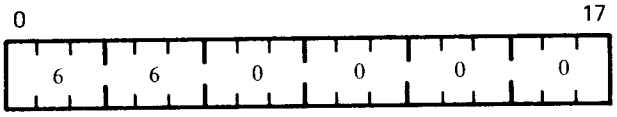
CLQ **Clear the Multiply Quotient Register**



Execute Time: 1.75 μs

The content of the multiply quotient register is replaced with 0.

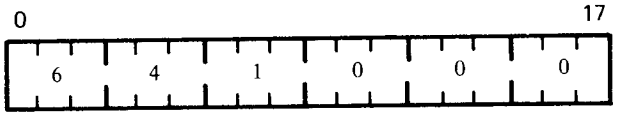
SHL **Shift Accumulator Bit Zero into the Link**



Execute Time: 1.75 μs

The content of accumulator bit 0 is shifted into the Link. The content of accumulator bits 1 through 17 are unchanged.

ECLA **Clear the Accumulator**



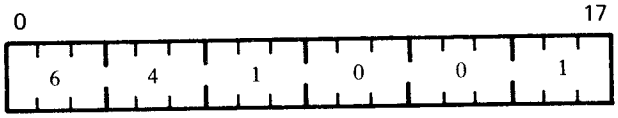
Execute Time: 1.75 μs

The content of the accumulator is set to 0. The Link remains unchanged.

Microcoding

The control instructions of the extended arithmetic element can be microcoded by ORing their instruction codes according to the chart in Figure 8-2. The most useful combinations are explained in detail.

LACS= OSC!ECLA **Load Accumulator From Step Counter**



Execute Time: 1.75 μs

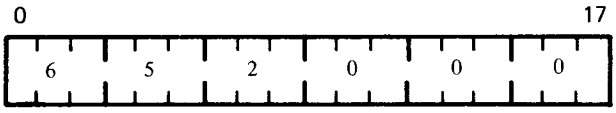
The accumulator is cleared and the content of the 6-bit step counter is loaded into accumulator bits 12 to 17.

√ = Valid Combination

Instruction Mnemonic							
OMQ							
CMQ	√	√					
ECLA	√	√					
AGS		√	√	√			
OAC	√	√	√	√	√		
CLQ	√	√	√	√		√	
SHAL	√	√	√	√	√	√	√
Instruction Mnemonic	OSC	OMQ	CMQ	ECLA	ABS	OAC	CLQ

Figure 8-2. EAE Control Instruction Microcoding

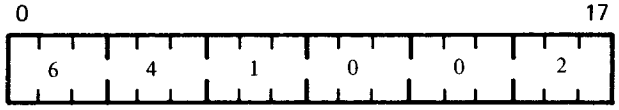
LMQ= **Load Multiply Quotient**
OAC!CLQ **Register from Accumulator**



Execute Time: 1.75 μs

The content of the multiply quotient register is replaced with the content of the accumulator. The accumulator remains unchanged.

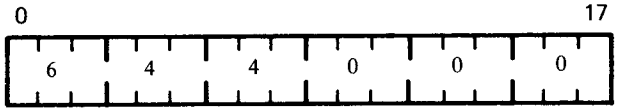
LACQ= **Load Accumulator From**
OMQ!ECLA **Multiply Quotient Register**



Execute Time: 1.75 μs

The content of the accumulator is replaced with the content of the multiply quotient register. The multiply quotient register remains unchanged.

GSM= **Get Sign and Magnitude**
ABS SHAL **of Accumulator**



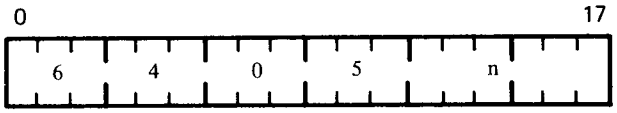
Execute Time: 1.75 μs

The content of accumulator bit 0 is entered into the Link, leaving the accumulator unchanged. Then, if the sign bit (bit 0) is a 1 (negative), the accumulator is 1's complemented.

An instruction which could prove useful in diagnostics is a combination of CLQ OAC ECLA OMQ. In this case, the content of the accumulator is loaded into the multiply quotient register, and then back to the accumulator. The content of the accumulator now reflects what was loaded into the multiply quotient register. The code is 653002.

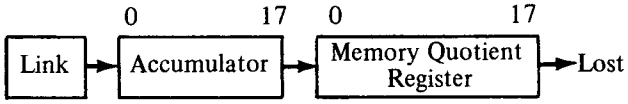
BASIC SHIFT INSTRUCTIONS

LRS n **Long Right Shift**

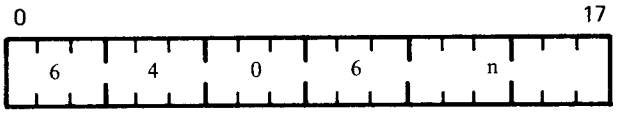


Execute Time: 2.75 + 0.125(n)
$0 \leqslant n \leqslant 36$

The accumulator and multiply quotient registers together function as a 36-bit shift register. Their contents are shifted n-bits to the right, where n is specified by the six low-order bits of the instruction. The step counter is automatically initialized to the 2's complement of n, and shifting stops when the step counter reaches 0. For each step, the content of accumulator bit 17 enters bit 0 of the multiply quotient register, and bit 17 of the multiply quotient register is lost. The content of the Link, usually initialized to 0, remains unchanged, but enters bit 0 of the accumulator at each step. The action is represented below:



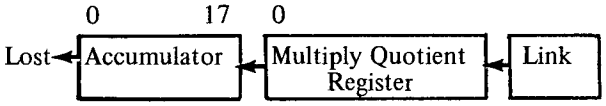
LLS n **Long Left Shift**



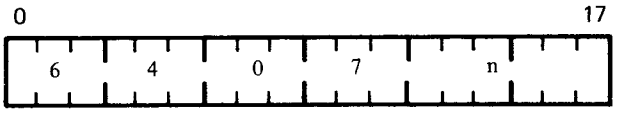
Execute Time: 2.75 + 0.125(n) μs
$0 \leqslant n \leqslant 36$

The accumulator and multiply quotient registers together function as a 36-bit shift register. Their contents are shifted n bits to the left where n is specified by the six low-order bits of the instruction. The step counter is automatically initialized to the 2's complement of n, and shifting stops when the step counter reaches 0. For each step, bit 0 of the memory quotient register enters bit 17 of the accumulator, the content of accumulator bit 0 is lost, and the content of the

Link, usually initialized to 0, remains unchanged but enters bit 17 of the multiply quotient register. Shifting is illustrated below.

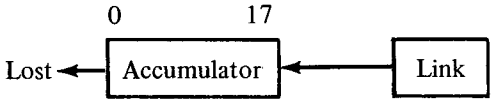


ALS n **Accumulator Left Shift**



Execute Time: $2.75 + 0.125(n)\ \mu s$
$0 \leqslant n \leqslant 18$

The content of the accumulator is shifted n positions to the left; where n is specified by the six low-order bits of the instruction word. Shifting stops when the contents of the step counter, automatically initializing to the 2's complement of n, reach 0. For each shift, the content of the Link, usually initialized to 0, enters bit 17 of the accumulator. The bits shifted out of accumulator bit 0 are lost. This action is illustrated below:



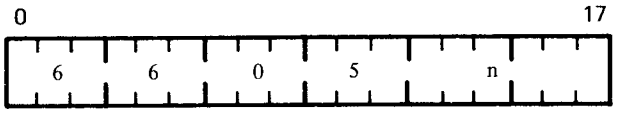
Microcoding

The shift instructions can be microcoded with certain functions to form more complex instructions. Figure 8-3 lists all possible combinations. The event time column indicates which function is carried out first during the execute cycle. Some of the most useful combinations are described below.

Instruction Mnemonic	√ = Indicates Valid Combination		
LLS			
ALS			
CLO	√	√	√
SHAL	√	√	√
ABS	√	√	√
OMQ	√	√	√
ECLA	√	√	
Instruction Mnemonic	LRS	LLS	ALS

Figure 8-3. Shift Instruction Microcoding

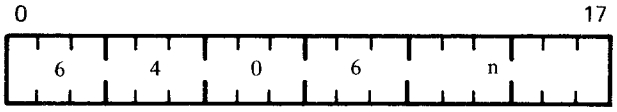
LASS n=LRSn SHAL **Long Right Shift Signed**



Execute Time: $2.75 + 0.125(n)\ \mu s$
$0 \leqslant n \leqslant 36$

The content of accumulator bit 0 enters the Link. Then a long right shift instruction is carried out. (Note that the content of the Link enters accumulator bit 0.)

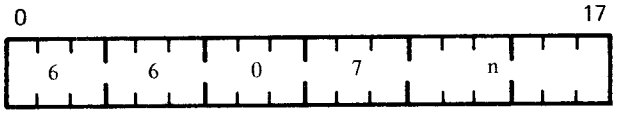
LLSSn=LLSn SHAL **Long Left Shift Unsigned**



Execute Time: $2.75 + 0.125(n)s$
$0 \leqslant n \leqslant 36$

The content of the accumulator bit 0 enters the Link. Then a long left shift instruction is carried out. Note that the content of the Link enters multiply quotient bit 17.

ALSSn=ALSn SHAL **Accumulator Left Signed**

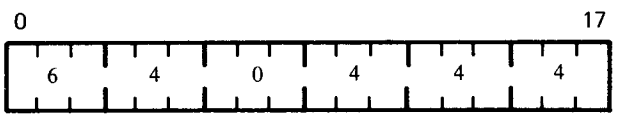


Execute Time: $2.75 + 0.125(n)$
$0 \leqslant n \leqslant 18$

The content of accumulator bit 0 enters the Link. Then a long accumulator left shift takes place. Note that the content of the Link enters accumulator bit 17.

NORMALIZE INSTRUCTIONS

NORM **Normalize**



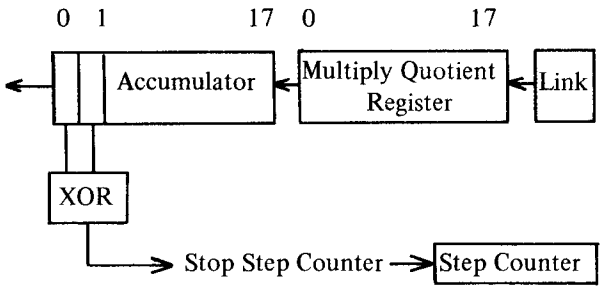
Execute Time: $2.75 + 0.125(n)\ \mu s$

where n is the number of shifts required to complete the normalize. The accumulator and multiply quotient registers together function as a 36-bit shift register. Their contents are shifted left until:

a. The content of accumulator bit 0 is different from the content of accumulator bit 1, or

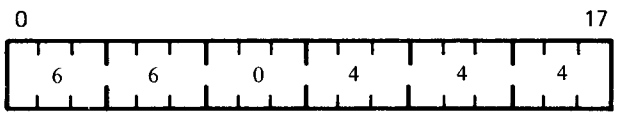
b. The content of the step counter reaches 0.

The content of the 6-bit step counter, which specifies the step count, is automatically initialized to 44_8 or 36_{10} by the content of the six low-order bits of the instruction. For each shift step, the content of bit 0 of the multiply quotient register enters bit 17 of the accumulator, and the content shifted out of accumulator bit 0 is lost. The content of the Link, usually initialized to 0, enters bit 17 of the multiply quotient register. If shifting halts because accumulator bit 0 does not equal bit 1, the step counter reflects the number of steps executed to reach that state. Its content (2's complement of step count plus the steps executed) is accessible with either the OSC or LACS instruction. The NORM operation is indicated below:



This normalize instruction can be combined with the control instruction SHAL, to form a normalize signed instruction.

NORMS **Normalize, Signed**



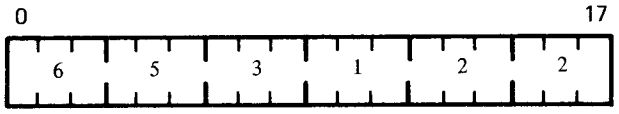
Execute Time: 7.25 μs

The content of accumulator bit 0 enters the Link. The accumulator remains unchanged. Then a normalize is carried out. Note that the content of the Link enters bit 17 of the multiply quotient register.

ARITHMETIC INSTRUCTIONS

Multiply

MUL **Multiply**

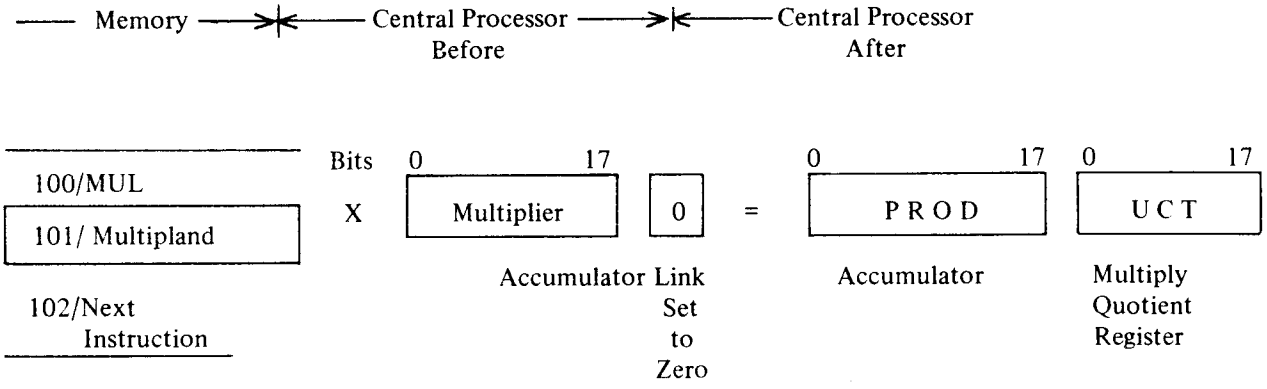


Execute Time: 7.0 μs

The content of the memory location which follows immediately after the MUL instruction is multiplied with the content of the accumulator. A double length (36-bit) product is formed in the accumulator and the multiply quotient register. The most significant 18-bits are contained in the accumulator, and the least significant bits in the multiply quotient register.

Prior to this instruction, the Link must be zeroed, the multiplier entered into the accumulator, and the multiplicand into the address following the MUL instruction.

At the beginning of a MUL execution, the multiplier is transferred to the multiply quotient register, the accumulator is cleared, and the step counter is initialized to the 2's complement of 22_8 (18_{10} steps) from the six low-order bits of the MUL instruction word. The execution, a multiplication of one unsigned quantity by another (18 bits, binary point of no consequence) halts when the step counter reaches 0. The content of the Link remains 0, the content of the register containing the multiplicand, the one following the MUL instruction, remains unchanged, and the program continues from the following instruction (two locations after MUL). The process is illustrated below:



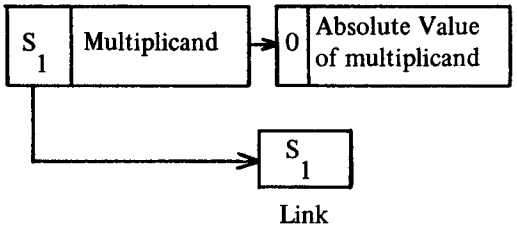
Signed multiplication can be carried out by using the MUL instruction with the ABS instruction and converting the multiplicand into its absolute value. This is done as follows:

Register	Content	
Y-6	LAC Multiplicand	Put the Multiplicand into the accumulator
Y-5	GSM	Stored sign in Link and Convert
Y-4	DAC Y	Store absolute value of multiplicand in Y
Y-3	LAC Multiplier	Load multiplier into accumulator
Y-2	ABS	Store sign in EAE sign bit and convert Multiplier to absolute value

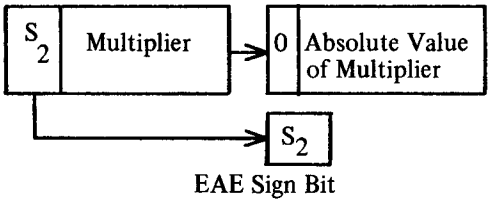
Y-1	MUL	Multiply two absolutes; 1's complement results if Link differ.
Y	Multiplicand	

The results are illustrated below

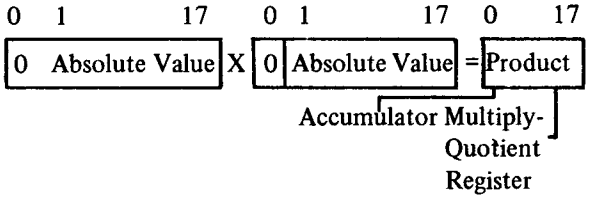
a. Memory Location Y Location Y



b. Memory Location Accumulator



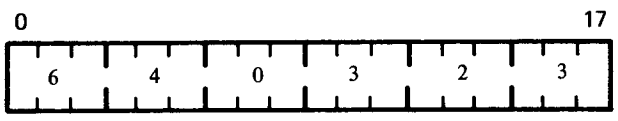
c.



If the EAE sign bit ≠ Link, 1's complement the product. Clear EAE sign bit in either case.

The MUL and ABS instructions can be microcoded to become the MULS instruction which is called Signed Multiply.

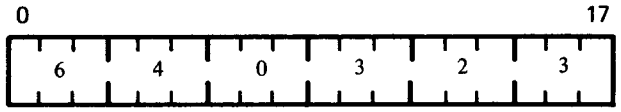
MULS **Signed Multiply**



Execute Time: 7μs

This is a combination of MUL and ABS. In signed multiplication, it is assumed that a GSM instruction has already been performed on the multiplicand. See explanation for MUL.

DIV **Divide Unsigned**



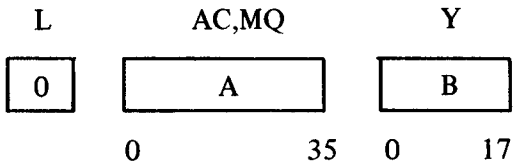
Execute Time: 7.25 μs

The content of the memory location Y which follows immediately after the DIV instruction divides the contents of the accumulator and multiply quotient register (an unsigned 36-bit dividend). The resulting quotient appears in the multiply quotient register, and the remainder is in the accumulator.

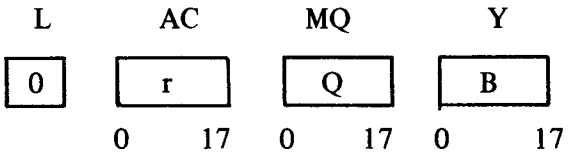
Prior to this instruction, the Link must be zeroed, and the dividend must be entered into the accumulator and multiply quotient register. This is done with a LAC (least significant half), LMQ, LAC (most significant half) sequence. If the divisor is not greater than the accumulator portion of the dividend, divide overflow occurs (the magnitude of the quotient exceeds the 18-bit capacity of the multiply/quotient register) and the Link is set to a 1 signaling this overflow condition. In this case, the data in the accumulator and multiply quotient registers is of no value.

A valid division halts when the step counter, initialized to the 2's complement of 23_8 (19_{10} steps) by the six low-order bits of the instruction, counts to 0. The divisor remains unchanged in core, and the program resumes at the next instruction - two after the DIV. The process is indicated below.

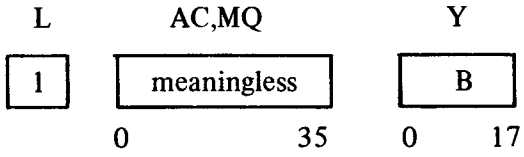
Pre-execution



Post-execution
(no overflow)



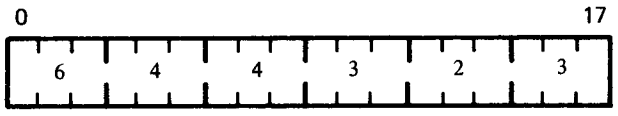
(overflow)



Instruction Sequence:

Memory Location	Contents
Y - 4	LAC Dividend (least significant
Y - 3	LMQ
Y - 2	LAC Dividend (most significant half)
Y - 1	DIV
Y	Divisor
Y + 1	Next instruction

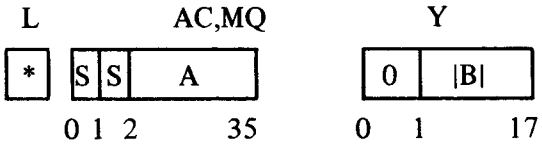
DIVS **Signed Division**



Execute Time: 7.25 μs

Divide the contents of the AC and MQ (a 36-bit signed dividend with the sign in AC_0 and bits 0 and 01, and the remaining 34 bits devoted to magnitude) by the contents of memory location Y (the divisor). The resulting quotient appears in the MQ with the algebraically determined sign in MQ_0 bit 0 and the magnitude (1's complement) in MQ bits 1 through 17. The remainder is in the AC with AC bit 0 containing the magnitude (1's complement). The address of Y is taken to be sequential to the address of the DIVS instruction word. The contents of Y are taken to be the absolute value of the divisor; the contents of the Link are taken to be the original sign of the divisor (DIVS assumes previous execution of an EAE GSM instruction). Prior to this DIVS instruction, the dividend must be entered in the AC and MQ (LAC of least significant half, LMQ, and LAC of most significant half). The MQ portion of a negative dividend is 1's complemented prior to the division. If the divisor is not greater than the AC portion of the dividend, divide overflow occurs (magnitude of the quotient exceeds the 17 bit plus sign capacity of the MQ), and the Link is set to one to signal the overflow condition; data in the AC and the MQ are of no value. A valid division halts when the step counter, initialized to the 2's complement of 23_8 (19_{10} steps), counts up to 0 (the six low-order bits of the DIVS instruction word specify the step count). The content of the Link is cleared to 0. The contents of Y are unchanged. The program resumes at the next instruction (memory location Y + 1).

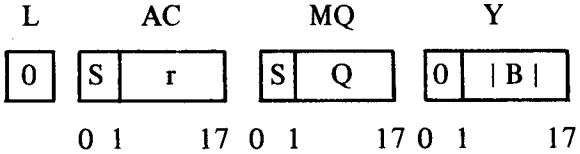
Pre-execution



*original sign of B

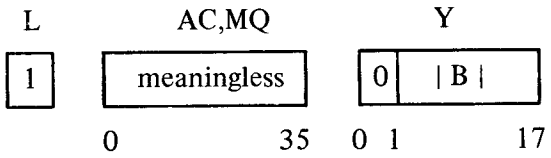
Post-Execution

(no overflow)



(S=Sign A) (S=Sign A ⊻ L)

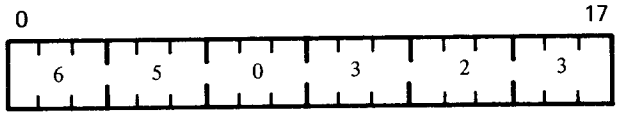
(overflow)



Instruction Sequence:

Memory Location	Contents
Y - 7	LAC Divisor
Y - 6	GSM
Y - 5	DAC Divisor in Y
Y - 4	LAC Dividend (least significant half)
Y - 3	LMQ
Y - 2	LAC Dividend (most significant half)
Y - 1	DIVS
Y	Divisor (absolute value)
Y - 1	Next Instruction

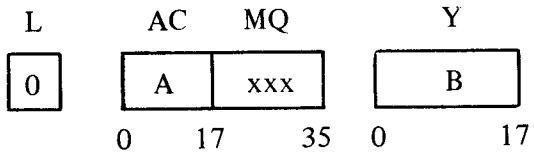
FRDIV **Fraction Divide Unsigned**



Execute Time: 7.25 μs

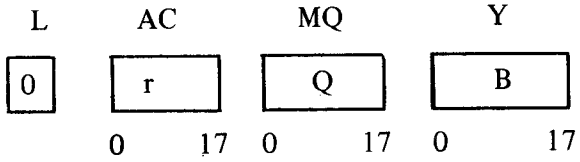
Divide the contents of the AC and the MQ (AC contains an 18-bit fractional dividend, MQ is zeroed at setup) by the contents of memory location Y (the divisor). The binary point is assumed at the left of AC (bit 0). The quotient appears in the MQ; the remainder is in the AC. The address of Y is taken to be sequential to the address of the FRDIV instruction word. Prior to this instruction, the contents of the Link must be 0, and the dividend must be entered in the AC (the set-up phase of FRDIV clears the MQ). If the divisor is not greater than the dividend, divide overflow occurs (magnitude of quotient exceeds the 18-bit capacity of the MQ), and the Link is set to 1 to signal the overflow condition; data in the AC and the MQ are of no value. A valid division halts when the step counter, initialized to 23_8 (19_{10} steps), counts up to 0 (the six low-order bits of the FRDIV instruction word specify the step count). The contents of the Link remain 0. The contents of Y are unchanged. The program resumes at the next instruction (memory location Y + 1).

Pre-Execution

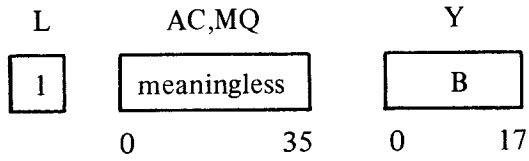


Post-Execution

(no overflow)

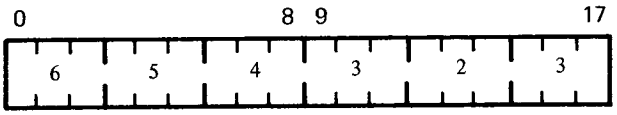


(Overflow)



Memory Location	Contents
Y - 2	LAC Divident
Y - 1	FRDIV
Y	Divisor
Y + 1	Next Instruction

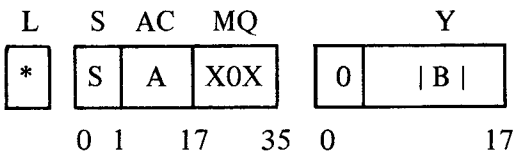
FRDIVS **Fraction Divide Signed**



Execute Time: 7.25 μs

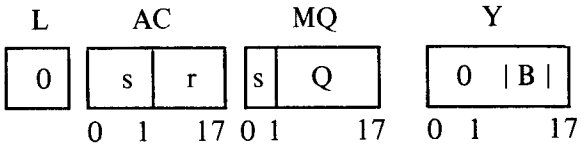
Divide the contents of the AC and the MQ (AC contains a signed fractional dividend, MQ is zeroed at set-up) by the contents of memory location Y (the divisor). The binary point is assumed between AC bit 0 and AC bit 1. The resulting quotient appears in the MQ with the algebraically determined sign in MQ bit 0 and the magnitude (1's complement) in MQ bits 1 through 17. The remainder is in the AC with bit 0 containing the original sign of the dividend and bits 1-17 containing the magnitude (1's complement). The address of Y is taken to be sequential to the address of the FRDIVS instruction word. The contents of Y are taken to be the absolute value of the divisor; the contents of the Link are taken to be the original sign of the divisor (FRDIVS assumes previous execution of an EAE GSM instruction). Prior to this FRDIVS instruction, the dividend must be entered in the AC (the set-up phase of FRDIVS clears the MQ and 1's complements the dividend, if negative, prior to the division). If the divisor is not greater than the dividend, divide overflow occurs (magnitude of the quotient exceeds the 18-bit capacity of the MQ) and the Link is set to 1, to signal the overflow condition. Data in the AC and the MQ are of no value. A valid division halts when the step counter, initialized to the 2's complement of 23_8 (19_{10} steps), counts up to 0 (the six low-order bits of the FRDIVS instruction word specify the step count). The contents of the Link are cleared to 0. The contents of Y are unchanged. The program resumes at the next instruction (memory location Y + 1).

Pre-Execution



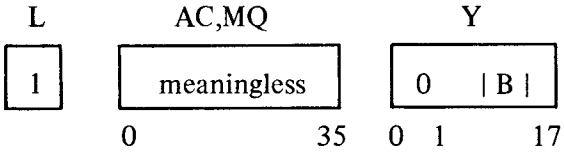
*original sign of B

Post-Execution
(No overflow)



(s = Sign A) (s = L ⊻ Sign A)

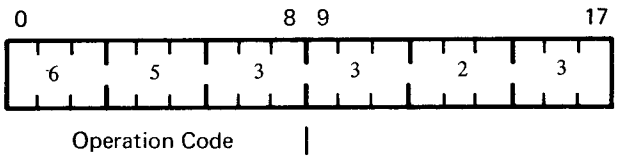
(Overflow)



Instruction Sequence:

Memory Location	Contents
Y - 5	LAC Divisor
Y - 4	GSM
Y - 3	DAC Divisor (absolute value) in Y
Y - 2	LAC Dividend
Y - 1	FRDIVS
Y	Divisor (absolute value)
Y + 1	Next Instruction

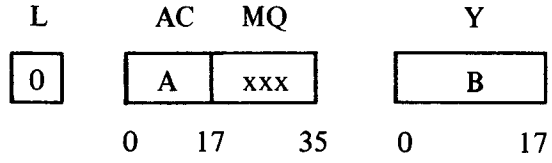
IDIV **Integer Divide Unsigned**



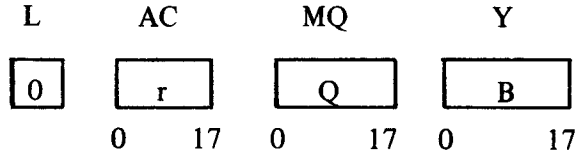
Execute Time: 7.25 μs

Divide the contents of the AC and the MQ (AC is 0, MQ contains a 18-bit integer dividend) by the contents of memory location Y (the divisor). The resulting quotient appears in the MQ; the remainder is in the AC. The address of Y is taken to be sequential to the address of the IDIV instruction word. Prior to this instruction, the contents of the Link must be 0, and the dividend must be entered in the AC (the set-up phase of IDIV transfers the dividend to the MQ and clears the AC). Division overflow occurs only if division by 0 is attempted; i.e., the quotient's magnitude will not exceed the 17-bit plus sign capacity of the MQ. The division halts when the step counter, initialized to the 2's complement of 23_8 (19_{10} steps), counts up to 0 (the six low-order bits of the IDIV instruction word specify the step count). The content of the Link is cleared to 0. The contents of Y are unchanged. The program resumes at the next instruction (memory location Y + 1).

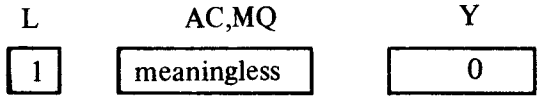
Pre-Execution



Post-Execution



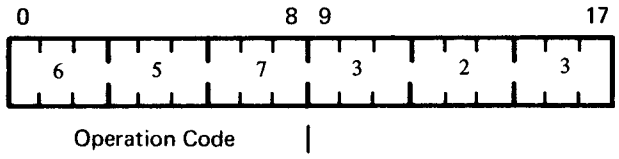
If Y = 0 (overflow)



Instruction Sequence:

Memory Location	Contents
Y - 2	LAC Dividend
Y - 1	IDIV
Y	Divisor
Y + 1	Next Instruction

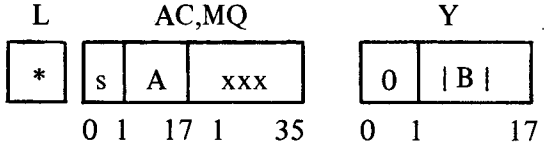
IDIVS Integer Divide Signed



Execute Time: 7.25 μs

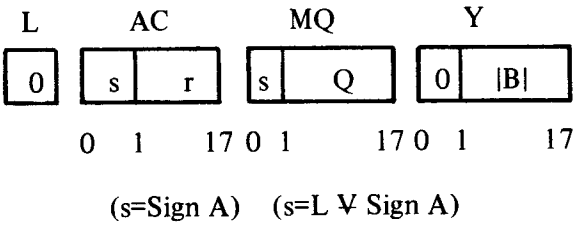
Divide the contents of the AC and the MQ (AC is 0, MQ contains a signed integer dividend by the contents of memory location Y (the divisor). The resulting quotient appears in the MQ with the algebraically determined sign in MQ in bit 0 and the magnitude (1's complement) in MQ bits 1-17. The remainder is in the AC with AC bit 0 containing the sign of the dividend and AC bits 1-17 containing the magnitude (1's complement). The address of Y is taken to be sequential to the address of the IDIVS instruction word. The contents of Y are taken to be the absolute value of the divisor; the contents of the Link are taken to be the original of the divisor (IDIVS assumes previous execution of an EAE GSM instruction).Prior to this IDIVS instruction, the dividend must be entered in the AC (the set-up phase of IDIVS transfers the dividend to the MQ, clears the AC, and 1's complements the MQ if the dividend is negative). Divide overflow occurs only if division by 0 is attempted; i.e. the quotient's magnitude will not exceed the 17-bit plus sign capacity of the MQ. The division halts when the step counter, initialized to the 2's complement of 23_8 (19_{10} steps), counts up to 0 (the six low-order bits of the IDIVS instruction word specify the step count). The contents of the Link are cleared to 0. The contents of Y are unchanged. The program resumes at the next instruction (memory location Y + 1).

Pre-Execution

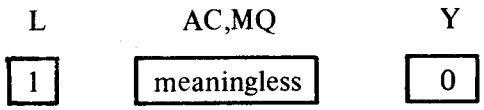


*original sign of B

Post-Execution



If Y = 0 (overflow)

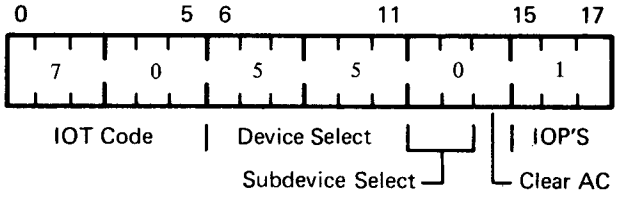


Instruction Sequence:

Memory Location	Contents
Y - 5	LAC Divisor
Y - 4	GSM
Y - 3	DAC Divisor (absolute value) in Y
Y - 2	LAC Dividend
Y - 1	IDIVS
Y	Divisor (absolute value)
Y + 1	Next Instruction

AUTOMATIC PRIORITY INTERRUPT INSTRUCTION SET

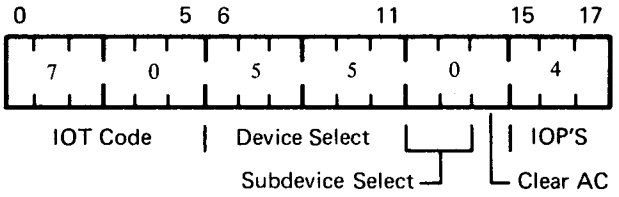
SPI **Skip on Priorities Inactive**



Execute Time: 2 μs

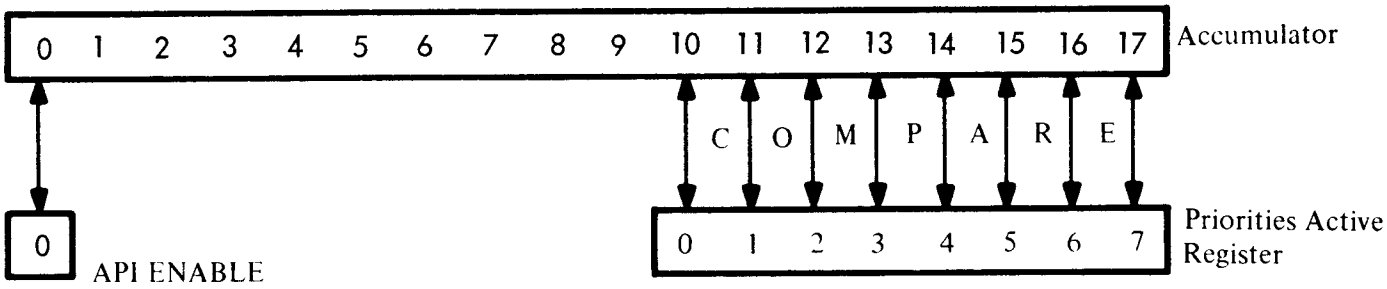
This instruction compares a condition code in the accumulator with part of the ENABLE bit and priorities active register. If any bit of the condition code matches the corresponding bit of the ENABLE or priority active register and both are set, the next instruction is skipped. Otherwise the next instruction is executed. The corresponding bits are shown below.

ISA **Initiate Select Activity**



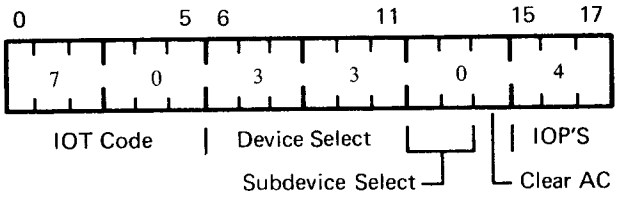
Execute Time: 4 μs

The content of accumulator bit 0 is placed into the ENABLE flip-flop; accumulator bits 6



through 9 are placed into bits 4 to 7 of the API request register, and accumulator bits 10 through 17 are placed into bits 0 through 7 of the API priorities active register. A diagram of this follows.

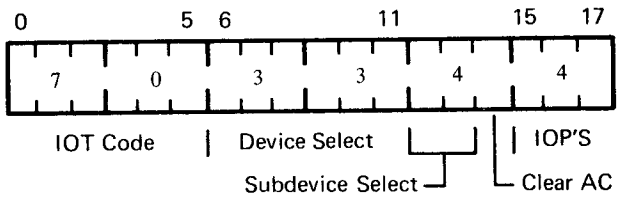
DBK **Debreak**



Execute Time: 4 μs

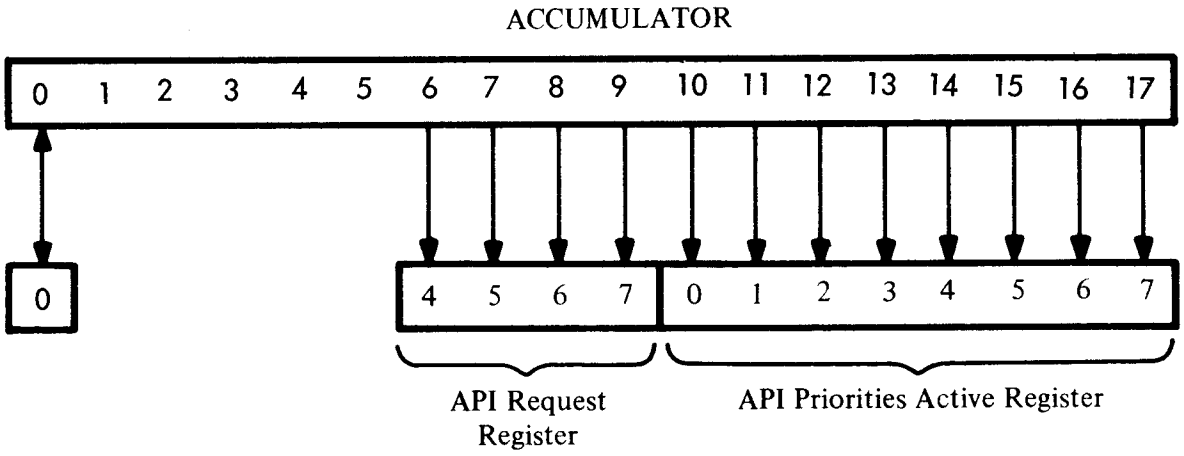
This instruction zeroes the highest priority presently in the decision register and moves the masking registers, thus clearing the way for future requests. It must be used to reset the priority active register after an SPA instruction has raised the status of an interrupt.

DBR **Debreak and Restore**



Execute Time: 4 μs

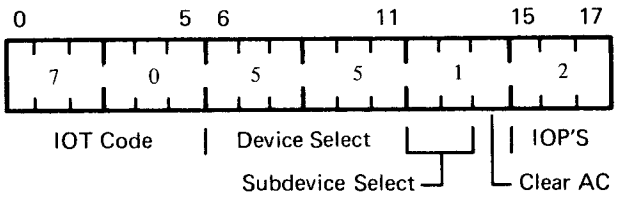
This instruction zeroes the highest priority presently in the priority active register, thus clearing the way for future requests. It also primes the PDP-15 to restore the Link, the program counter and memory protect mode to their status at the time the API request was honored. The actual restoration occurs at the execution of the JMP I instruction exiting the subroutine which must immediately follow the DBR instruction.



NOTE

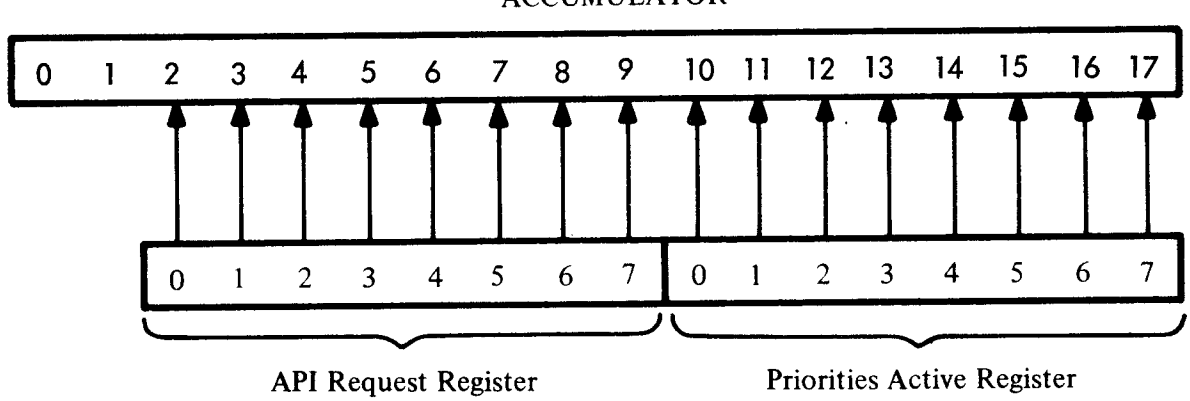
The ISA and SPI instructions can be microcoded to 705505. This instruction is used to first test that the program segment currently in progress is not already masked to the requested priority level, and then, if not, to initiate a masking to the requested level.

RPL **Read Priority Levels**



Execute Time: 4 μs

The content of the API request register is read into accumulator bits 2 to 9, and the content of the priorities active register is read into accumulator bits 10 to 17, as shown below.

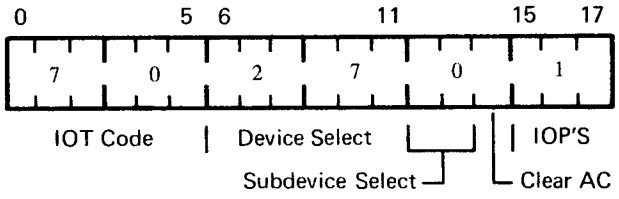


MEMORY PARITY

The option adds a nineteenth bit to each of the words in a 4096 word core-memory module. These bits retain the parity indication for their associated words. Parity is generated when a word is entered in memory and checked when the word is read out of its memory location. Detection of a parity error can initiate a program interrupt or half execution of the program, as the programmer selects. Systems with memory parity have a 1 μs memory cycle time. If the memory protection or the relocation and protect option is added to a system with parity these operations occur in parallel and would not be additive. For parity and memory protection the cycle time remains at 1 μs, for parity and relocation and protect the memory cycle is 1.025 μs.

The IOT instructions associated with memory parity are as follows:

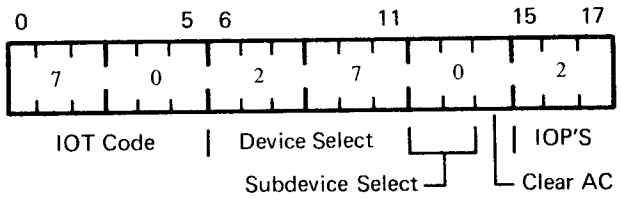
SPE **Skip on Parity Error**



Execute Time: 2 μs

If the parity error flag is posted, the program counter is incremented by 1 and the next instruction is skipped.

CPE **Clear Parity Error**



Execute Time: 4 μs

The parity error flag is cleared to 0 when this instruction is executed.

REAL-TIME CLOCK

The real-time clock, when enabled, counts in memory location 00007 the number of cycles completed by any one of three inputs:

a. The line voltage (50 or 60 Hz)

b. An M405 R-C Clock which can be set to any frequency from 0 to 10 kHz

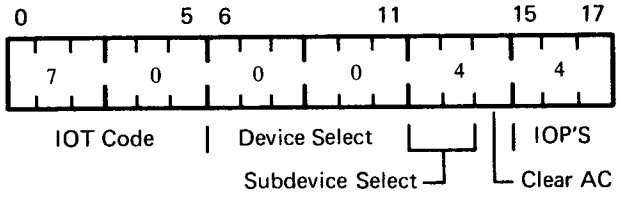
c. A user supplied TTL compatible signal which can be fed through a coaxial cable to a BNC connector on the PDP-15.

When location 00007 overflows an interval program interrupt or API request, if available, is generated informing the monitor that its preset interval is over. The monitor must either disable the clock or reinitialize location 00007 to the 2's complement of the number of counts it needs to tally.

The incrementing of location 00007 during a real-time clock request occurs via the I/O processor, using its increment-memory facility. A real-time clock request takes priority over API, PI and IOT requests.

The following IOT instructions are provided for use with the clock:

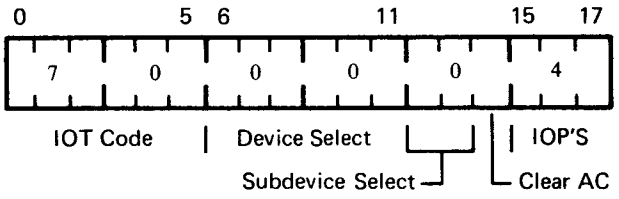
CLON **Clock On**



Execute Time: 4 μs

The real-time clock is enabled to begin incrementing location 00007, and its flag is cleared.

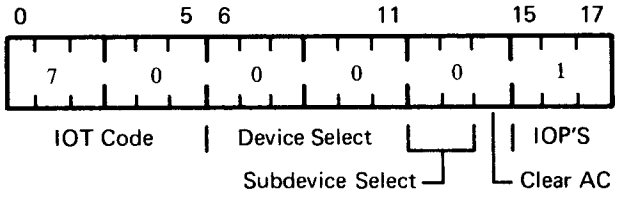
CLOF **Clock Off**



Execute Time: 4 μs

The real-time clock is disabled, preventing it from incrementing location 00007.

CLSF **Skip On Clock Flag**



Execute Time: 2 μs

The program counter is incremented and the next instruction skipped if the clock flag is set.

While the facility is enabled, requests for clock breaks have priority of acceptance over API and PI requests. The first clock break may occur at any time up to 17 ms after the facility has been enabled. Subsequent breaks occur at the clock rate (60 or 50 times per second).

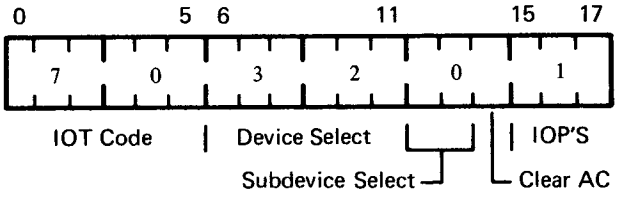
POWER FAILURE

The PDP-15 contains circuitry which provides optimum protection of programs during machine turn-on or turn-off, whether accidental or intended. In the basic machine, the circuitry is designed to protect the contents of memory. However, the addition of the system power auditor, an option of the I/O Processor, further allows time to store active registers before the system stops during a power failure, and a system restart and the subsequent restoration of these registers when system power is reapplied.

The basic PDP-15 is not affected by power interruptions of less than 15 ms duration. Active registers in the processor (AC, AR, PC, etc.) will lose their contents when interrupts of longer duration occur but memory will not be disturbed. The power failure protection option provides for saving the contents of active registers in the event of longer power interrupts and for automatic restart of the system when power is restored. The restart feature is switch-selected by the operator to be enabled or disabled. When enabled, the program in progress resumes execution at location 00000. The system must be operating with the program interrupt facility (or the AID) enabled to sense the option's initiation of a program interrupt to save the register contents at the time of the line power failure. If API is available the power failure option interrupts on its highest level and uses memory address 52.

There is only one instruction associated with the Power Failure Option. That is:

PFSF **Skip On Power Low Flag**



Execute Time: 2 μs

The state of the power low flag is tested, and if set, indicating that system line voltage has dropped and that this flag has posted an interrupt, then the reset instruction is skipped. The flag is cleared by the power clear signal when the power interruption is over.

MEMORY PROTECTION OPTION

The memory protection feature establishes a foreground/background environment for PDP-15 processing activity by specifying the boundary between protected (lower) and unprotected (upper) regions of system core memory. Allocation of memory locations (in increments of 256 words) to the protected region is dynamic and program controlled. A boundary register, added by the option, stores the location of the upper limit of the protected region. It is loaded from bits 3 through 7 of the AC by a MPLS instruction.

The KM15 monitors the instruction about to be executed, and the protect feature transfers control to a monitor program should the instruction be in the category of "illegal instructions", before the instruction is executed. If a program tries to reference a nonexistent memory bank, the KM15, if it has been enabled, transfers control to the monitor program. The protect feature increases all memory cycle times by 100 ns and write cycles in user mode by an additional 100 ns.

The memory protect (or user mode) may be enabled either by programmed instruction, or by depressing the PRTCT switch on the console, and pressing the START key. When enabled, the option will trap the following:

IOT	Input/Output
CAF	Clear All Flags
XCT of XCT	Chained Execute Instructions
HLT	Halt
OAS/LAS	Load AC From Data Switches

References to nonexistent memory

References to locations below the boundary limit

Trapping causes the execution of an effective JMS instruction after the machine cycle that attempts to violate. The address referenced by the effective JMS instruction will be either location absolute 20, if the program interrupt facility is disabled, or location absolute 0, if the program interrupt facility is enabled. The violation flag is set.

The nonexistent memory flag is also set if the violation was caused by a program or I/O Processor reference to nonexistent memory. A single cycle I/O processor reference to nonexistent memory probably will be detected by the KM15.

User mode is disabled in the following ways:

I/O RESET Key

The detection of a violation

CAL Instruction (which never causes a violation)

A Program Interrupt

An API Interrupt

If user mode is enabled when an API break starts, and the API channel address contains a HLT, OAS, or IOT - rather than the normal JMS - that instruction will be inhibited, user mode will be disabled in the normal fashion and no violation will be detected.

If user mode is disabled when a reference to nonexistent memory is made, the nonexistent memory flag is set, no trap occurs, and the program continues after a 1 μs pause. If a reference to nonexistent memory occurs when user mode is enabled, the violation flag is also set and the trap occurs.

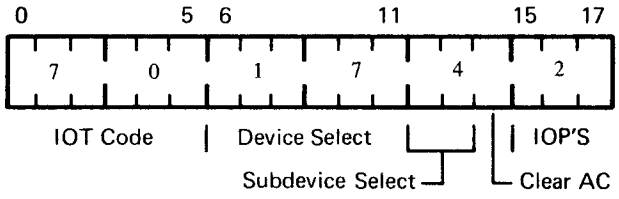
HLT, OAS and IOT instructions are totally inhibited when the memory protect option is enabled. If the HLT or OAS is combined with any other operate group instruction (microprogramming), the other parts of the operate group instruction are executed before the trap. (The exception is SKP which is not executed.) The second XCT in a chain of XCT instructions is trapped before execution.

The state of the protect mode (a 1 for user mode) is stored in bit 1 of the storage word by the operations that save the state of the machine (CAL, JMS, PI). The stored PC will contain one more than the location of the violating instructions, except for JMP to a protected area. In this case, the stored PC will contain the protected address.

The sole operator control is the PRTCT switch, which has an indicator above it. This indicator lights when in user mode. The PRTCT switch is used in conjunction with the START key to establish the proper mode at the beginning of program execution. If the switch is up, then the program is started in user mode. The switch has no further effect.

The IO RESET key clears the boundary register, violation and nonexistent memory flags, and user mode (i.e., memory protect is turned off).

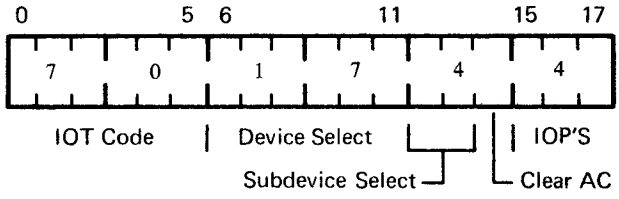
MPEU **Enter User (Protect) Mode**



Execute Time: 4 μs

Memory protect mode will be entered at the end of the next instruction that is not an IOT.

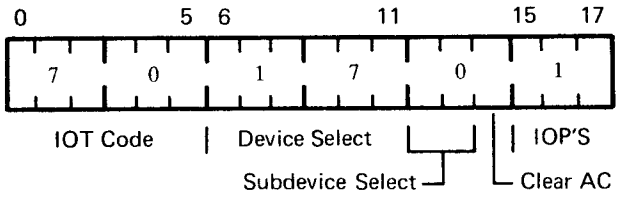
MPCNE **Clear Nonexistent Memory Flag**



Execute Time: 4 μs

The nonexistent memory flag posted when nonexistent memory has been referenced, is cleared by the IOT.

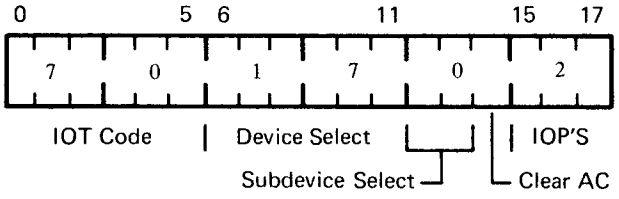
MPSK **Skip On Violation Flag**



Execute Time: 2 μs

The memory protect violation flag will be set whenever the execution of an instruction has violated the provision of memory protection (see above).

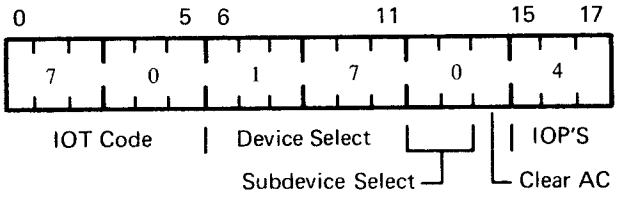
MPCV **Clear Violation Flag**



Execute Time: 4 μs

The violation flag, set if the boundary has been violated or an illegal instruction attempted, is cleared by this IOT.

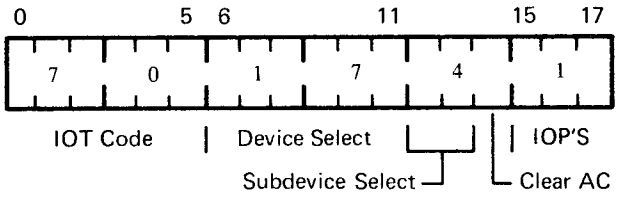
MPLD **Load Memory Protect Boundary Register**



Execute Time: 4 μs

Load the memory protection register with the contents of AC 3 through 7. The boundary register will store the number of 1024 word blocks to be protected.

MPSNE **Skip On Nonexistent Memory Flag**



Execute Time: 2 μs

The nonexistent memory flag is set whenever the processor attempts to reference a nonexistent area of core. For a 32,768 word machine, the flag never gets set.

MEMORY RELOCATION AND PROTECT - KT15

The relocation and protect hardware will consist of two registers. The relocation register and the core allocation register. The relocation register provides the lower limit of the user program and relocates the user upward from real machine-location 0 by the quantity contained in the register. The core allocation register assigns the quantity of locations available to the user. Any attempt by the user to reference core locations exceeding the limit of the core allocation register will cause a protect violation. Memory relocation and protect adds 225 ns to each CPU memory reference cycle.

There are two modes of operation in the PDP-15 protect system: user mode and monitor mode. In monitor mode, the relocation and protect hardware is disabled and the machine functions as it would without protect hardware. The program running in monitor mode addresses real locations within the system. In user mode, the relocation and protect hardware is enabled. The machine is programmed as though the user had a machine all to himself. His memory begins from location 0 and goes up to the content of the core allocation register. In the real machine, the program is located from the contents of the relocation register up, however, with the excep-

tion of I/O operations and the CAL instructions; the user has the machine virtually to himself and the programs that way. The following special cases occur in user mode.

CAL Instruction

When in user mode, and the CAL is given, the user mode is disabled, (monitor mode envoked), the CAL goes to location 20 in the real machine (not the virtual or relocated machine). The virtual program counter is saved in real 20, and real 21 is executed.

Program Interrupt

When a program interrupt is given, the user is disabled, the virtual PC is saved in real location 0, and location 1 is executed.

Automatic Priority Interrupt

When in user mode, an API causes monitor mode to be entered, and an instruction in the real machine specified by the I/O device is executed. This instruction will be a JMS to the device handler. The device handler will run in monitor mode. The device handler entry will receive the virtual program counter.

Data Channel

Data Channel operations will never be relocated.

Real-Time Clock

The real-time clock will always increment location 7 in the real machine; attempts to reference the contents of location 7 in the real machine must be handled through the monitor (with a CAL instruction).

Auto Increment Registers

Each user will have a complete set of auto-increment registers located at locations 10 to 17 of the user's virtual machine. In addition, the monitor can utilize the auto-increment registers in 10 to 17 of the real machine in monitor mode.

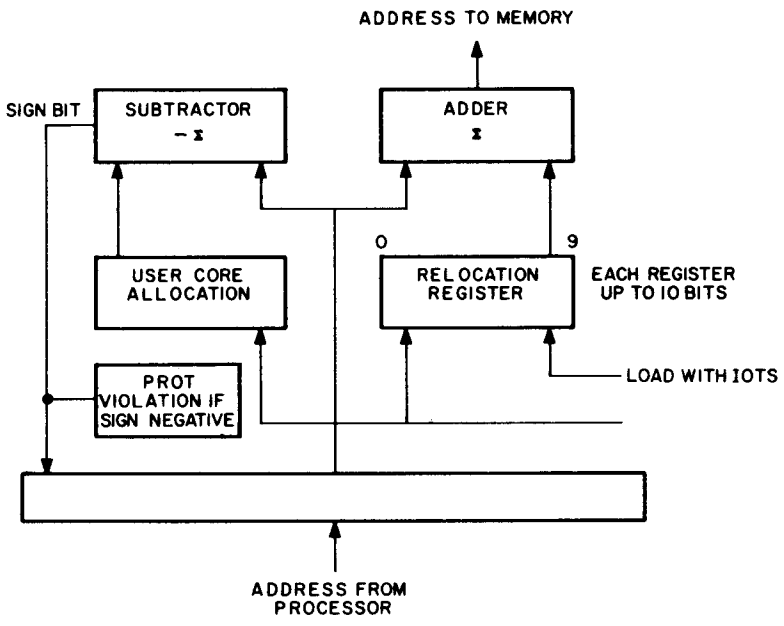


Figure 8-4. Hardware

DBR Instruction

The debreak and restore instruction (DBR) is used when returning from monitor mode to user mode. The instruction primes the PDP-15 to return the Link and the program counter to their status in user mode. The protect bit, if set, will initiate relocation. Since CAL, PI and API JMS save the virtual PC (not the real PC), the set protect bit causes return to the correct location in real memory, the virtual PC is restored, and user mode is invoked.

THE HARDWARE

The hardware involves a box between the processor and memory which computes relocated addresses and detects protect violations (see Figure 8-4).

Two registers of up to 18 bits are utilized in this scheme. One register is the relocation register whose contents are added to each processor-supplied address when in user mode. The user core-allocation register is subtracted from each processor address and if the result is negative, a violation has occurred. The relocation register is initialized to the user's lower address and the core allocation register is initialized to the quantity of core assigned to the user. For example, if a user were to need 4K of core, and the monitor determined the first available location was location 7301_8; the relocation register would be loaded with 7301_8; and the core allocation register with 10000_8.

CHAPTER 9
PERIPHERAL OPTIONS

A wide range of peripheral equipment is available to expand the capabilities of all PDP-15 systems.

STANDARD INPUT/OUTPUT DEVICES

High-Speed Paper Tape Reader and Punch, Type PC15

Standard on PDP-15/20, 15/30, and 15/40 systems. The perforated paper-tape reader can photoelectrically sense 8-channel paper tape at a rate of 300 characters-per-second. Under program control, data may be read in either alphanumeric (one character) or binary (three character) mode. The use of a paper-tape reader buffer and buffer-full flag permits the continuation of processing during the reading functions.

The 50 character-per-second paper-tape punch is mounted on the same chassis as the reader. A single output instruction causes an 8-bit character to be transferred from the PDP-15 accumulator to a punch buffer, from which it is punched on the tape. Fanfold paper tape is normally used with the paper-tape reader and punch.

Card Reader and Control, Type CR03B

The CR03B Card Reader provides the PDP-15 with a punched-card data input facility. It reads standard 80-column, 12-row punched cards at a rate of up to 200 cards per minute. Solid-state circuit design and continuous status checks ensure system reliability. The output hopper can store 450 cards in their original deck orientation.

MASS STORAGE DEVICES

DECtape

The DECtape system provides a unique fixed-address magnetic tape facility for program and data storage and retrieval. In environments where long batch-processing queues are not frequently required, DECtape is an excellent

substitute for punched cards. A single pocket-sized reel of DECtape can store 150,000 18-bit words; more information than a deck of 5,000 cards. At the same time, it costs considerably less than a reel of IBM-compatible magnetic tape. This compact, inexpensive format allows each user to have his own personal library of programs and data files on a pocket-sized reel, easily mounted on the transport in less than 15 seconds.

DEC*tape features*:

Fixed-position addressing to permit the selective reading or updating of information without the necessity of reading or rewriting the entire block.

Automatic word transfer, via the PDP-15 data channel facility, to allow concurrent processing and data transfer.

Bidirectional operation to allow reading, writing, and searching in either direction.

Redundant phase recording which insures transfer reliability, reduces the problem of skewing and minimizes bit dropouts.

Prerecorded timing and mark tracks to simplify programming and to permit block and word addressability.

DECtape Control, Type TC15 - The DECtape Control, Type TC15, controls up to eight DECtape transports, Type TU55. Binary information is transferred to and from the PDP-15 at the rate of one 18-bit word every 200 μs, using the data channel facility. Mode of operation, function, and direction of motion are controlled by status registers which can be loaded and read by the computer.

DECtape Transport, Type TU55 - The DECtape Transport, Type TU55, provides bidirectional reading and writing of DECtape reels. Each 3-in. diameter reel can hold 3,000,000 bits of information (over 150,000 18-bit words) recorded at 375 bpi. Tape moves at 80-ips and requires no vacuum column pinch rollers, or capstans.

Magnetic Tape Systems

PDP-15 offers both 7- and 9-channel IBM-compatible magnetic tape systems. Transports are currently available to operate at either 45 or 75 ips and at any of three recording densities.

Automatic Magnetic Tape Control, Type TC59 - The Automatic Magnetic Tape Control, Type TC59, transfers data to and from IBM-compatible transports via the data channel facility. Up to eight transports can be handled by a single control, and both BCD and binary modes are available. One TC59 control can handle both 7- and 9-channel transports at both 45 and 75 ips. Read/write functions, recording density, and tape manipulation functions are controlled by status registers which can be loaded and read by the PDP-15.

Magnetic Tape Transports, Type TU20, TU20A, TU30, TU30A - The Type TU20 Magnetic Tape Transport can read and write 7-channel IBM-compatible tapes at 45 ips and 200, 556, or 800 bpi. One 18-bit PDP-15 word is written as three tape characters on the Type TU20.

Its 9-channel counterpart, the Type TU20A Magnetic Tape Transport, operates at the same speed at 800 bpi. In two-character mode, the TU20A reads or writes two 8-bit characters per 18-bit word (ignoring two bits), while in three-character mode it reads or writes three 6-bit characters (one PDP-15 word) as three 8-bit tape characters.

The TU30 and TU30A Magnetic Tape Transports are respectively 7- and 9-channel units that operate at 75 ips.

Disk Systems

DECdisk, Type RF15/RS09 - The Fixed Head Disk System, Type RF15/RS09 is a new bulk storage medium for use with the PDP-15. The basic system consists of one RF15 Control unit and one RS09 Serial Disk unit, providing a bulk storage capacity of 262,144 18-bit words. Seven additional disk units can be accommodated by

the control unit, increasing storage capacity to 2,097,152 words. Data transfers ranging from one word to 32,768 words are performed via the multi-cycle data channel facility. Addressing is by disk unit, track and word.

Format Specifications

Storage Capacity (18-bit words) -	262,144
expandable to -	2,097,152
Number of Tracks	128
Words Per Track	2,048

Timing Specifications **Power**

	60Hz	50Hz
Average Access Time	16.7 ms	20.0 ms
Minimum Access Time	250 μs	250 μs
Worst-Case Access Time	33.3 ms	40.0 ms
Word-Transfer Rate		
High	62.5k wd/s	50k wd/s
Medium	31.2k wd/s	25k wd/s
Low	15.6k wd/s	12.5k wd/s

Disk Pack and Control, Type RP15/RP02 - The RP15 controller interfaces the Disk Pack drive to the PDP-15. Transfers are made through a single-cycle data channel. Up to eight RP02 drives can be handled on the same control. The total capacities of the RP02 drives are 10.24 million words. The total bulk storage capacity with eight RP02 drives is 81,920,000 18-bit words. Average transfer rate with the RP02 drive is 135k wd/s. Average access time, including the rotational latency time of 12.5 ms is 62.5 ms.

LINE PRINTERS AND PLOTTERS

Automatic Line Printer, Type LP15

The Type LP15 Automatic Line Printer is available in two models: The Type LP15A is a 300 line-per- minute unit. The Type LP15C is a 1000 line-per-minute unit. Both models print text in lines of up to 132 characters from a 64 character set.

Incremental Plotter and Control, Type XY15

The Type XY15 Incremental Plotter and Control uses CalComp Model 563 or 565 plotters at rates of 12,000 or 18,000 points-per-minute. Paper width is either 31-in. (Model 563) or 12 in. (Model 565) and plotting increments of 0.005 and 0.01 in. are available.

DATA COMMUNICATIONS DEVICES

Multi-Station Teletype Control, Type LT19

The Multi-Station Teletype Control is available to interface from one to five communications terminals to the PDP-15 Central Processor. (See Figure 9-1.)

Teletype Control, Type LT19A - The LT19A can control up to five Teletype Line Units, Type LT19B.

Line Units, Type LT19B - LT19B Line Units are full-duplex Teletype interfaces for either KSR or ASR models: each LT19B unit handles one Teletype or LT19C Adapter.

EIA Adapters, Type LT19C - The addition of the EIA Adapter, Type LT19C, to an LT19B unit allows the line unit to communicate with low-speed data sets.

Teletype Control, Type LT15

The Teletype Control, Type LT15 is standard on the PDP-15/30 and PDP-15/40 Systems. It allows the addition of a second Teletype unit for use in the Background/Foreground environment.

Console Teleprinter and Control - The console teleprinter, Teletype Model KSR35, is standard on the PDP-15/20, 15/30, 15/40 (ASR33 on the

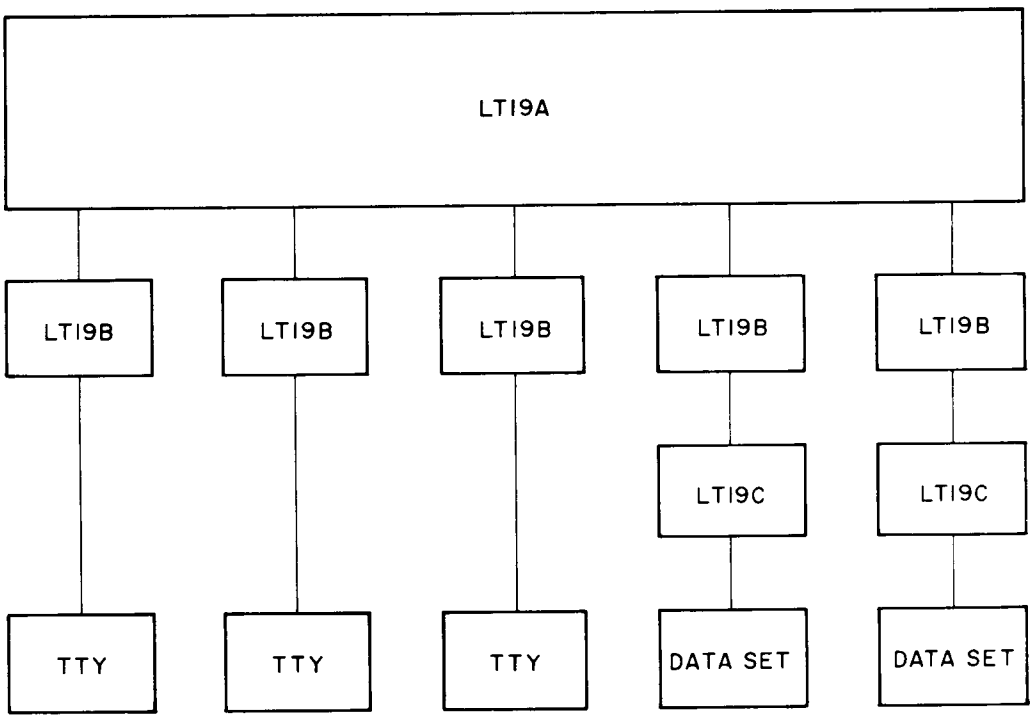


Figure 9-1. Multi-Station Teletype Control

PDP-15/10). These Teletypes can be used to type-in or print-out information at rates up to 10 cps. The keyboard control includes an 8-bit buffer to hold the last character (ASCII code) struck on the keyboard and a flag to signal the processor of the presence of a character, while the printer control contains an 8-bit buffer to hold one character while it is being printed. When the Model ASR33 Teletype is used as the console teleprinter, its paper-tape reader and punch are interfaced to the PDP-15/10.

Data Communication System Type DP09A

The Bit-Synchronous Data Communication System, Type DP09A provides interface facilities between a PDP-15 and a bit-serial communication device. The DP09A serializes the characters for transmission, and assembles the serial stream into characters for reception. Operation is full duplex. Both the receive and transmit sections are double-buffered to permit one full character transmission time for loading or reading the DP09A.

DISPLAY DEVICES

VP15 Display Family

Three low-cost, point-plotting display systems can be provided through the use of the VP15 family of display controllers. The various controller/display device combinations allow the selection of a system particularly suited to a given user application.

Storage Tube Display, Type VP15A - The Type VP15A is a unique and extremely useful point-plotting display terminal which uses a VT01 Storage Tube Display. Points are plotted on a 1024 by 1024-bit matrix on the 5-1/4 by 6-3/8 in. CRT. Two IOT-selectable modes of opera-

tion, store and non-store, are provided. In the store mode of operation, plotted points remain visible up to 15 minutes. No refresh memory is required. In the non-store mode of operation, a point-refreshing rate of at least 30 cps is required to keep points visible, but a faster response time is achieved. In either case, only one intensity level is provided.

Oscilloscope Display, Type VP15B - The Type VP15B Oscilloscope Display provides a point-plotting capability at an extremely low cost. The 1024 by 1024-bit raster is displayed on the Tektronix RM503 X/Y Oscilloscope. The RM503 has various intensity levels and can be used with a light pen and control, Type VP15BL.

Incremental Display, Type VP15C - The VP15C version of the basic VP15 display controller is for use with the VR12 display. This provides the user with a display device with useful display dimensions of 7 x 9 in. The VP15C provides various intensity levels.

INTERPROCESSOR BUFFER SYSTEMS

Two interprocessor models are available to facilitate inter-computer communications.

Interprocessor Buffers, Type DB99, DB98

There are two models of the interprocessor buffer systems. These are established by the type of programmed data processor interfaced with the system and are the DB99 Interprocessor Buffer System and the DB98 Interprocessor Buffer System.

The Model DB99 Interprocessor Buffer System is used to interface two PDP-15 Programmed Data Processors. The Model DB98 Interprocessor Buffer is used to interface a PDP-15 Programmed Data Processor with a PDP-8 Programmed Data Processor.

The interprocessor buffer system permits one data processor to communicate with a second data processor and to transfer data between accumulators or between memories of the interfaced data processors. Data transfers between accumulators occur on a word-by-word basis and are called program-controlled transfers. Data transfers between memories occur on a block-by-block basis and are called data-channel transfers. These use the multi-cycle data channel facility.

ANALOG-TO-DIGITAL CONVERTERS

Digital supplies analog-to-digital conversion systems with high speed and wide dynamic ranges for up to 1000 channels. Both single-ended and differential systems are available, and amplifier and sample-and-hold options can be provided.

Analog-to-Digital Converter and Multiplexer, Type AF01B

The Analog-to-Digital Conversion System, Type AF01B, includes a high-speed successive approximation converter with switch-selected word length from 6 to 12 bits. Conversion time varies with word length from 9 to 35 μs and error from ±.025%. The multiplexer handles up to 64 single-ended inputs and analog signal ranges from +10 to -10V. The AH03 Amplifier and AH02 Sample-and-Hold Devices may be used between the converter and multiplexer, while the Type AC01B Sample-and-Hold System may be used before the multiplexer. Each AC01B accommodates up to eight channels.

Integrating Digital Voltmeter System, Type AF04B

The Type AF04B Integrating Digital Voltmeter provides an automatic-ranging integrating digital voltmeter and low-level scanner that can handle analog signals with full scale values from 10 mV to 300V. Resolution is to 10 mV. Designed for three-wire inputs, the Type AF04B can expand up to 1000 channels.

DIGITAL-TO-ANALOG CONVERTERS

Digital-to-analog converters with 12-bit accuracy are available packaged in groups of three or, for large numbers, with a D/A controller.

Digital-to-Analog Converter Control, Type AA05B

The Type AA05B Digital-to-Analog Converter Control permits the multiplexed control of up to 64 digital-to-analog converter channels. Random addressing is used, and a channel may be addressed and updated by a single PDP-15 command.

Sample-and-Hold Option, Type AH02

The AH02 Sample-and-Hold option consists of a sample-and-hold amplifier capable of tracking a full scale excursion in 12 μs to 0.025% accuracy. In the hold mode, the droop (decay) is less than 1 mV/ms.

OPERATIONAL AMPLIFIER, TYPE AH03

The AH03 Operational Amplifier allows for a number of input voltage ranges between ±15V. Open loop gain is 2×10^6, unity gain frequency response is 10 mHz and the input independence between inputs is 6 mΩ.

I/O BUS ADAPTER, TYPE DW15

The DW15A I/O Bus Adapter converts a positive PDP-15 I/O bus of +2.5V and ground signals to a negative bus of -3V and ground. This adapter interfaces PDP-9 peripherals to PDP-15 Systems.

CHAPTER 10
PDP-15 CONSOLE

The PDP-15 console (see Figure 10-1) provides access to, or control over, virtually every portion of the PDP-15. All registers, buses, and controls are multiplexed down a single console cable to display the equivalence of 24 registers and 34 control functions. The console provides extensive control for real-time debugging and comprehensive man-machine interaction during checkout.

INFORMATION/CONTROL SWITCHES

The following two-position rocker switches are on the console keyboard:

Number	Designation	Number	Designation
18	Data	1	Repeat
15	Address	1	Halt Loop
1	Clock	1	Single Time
1	Register Group	1	Single Step
1	Parity Error	1	Single Instruction

Data

These 18 switches may be read indirectly into the accumulator by the execution of a OAS (OR accumulator content with switches content) instruction in an operating program. Data may be inserted manually into the machine with these switches by means of the DEPOSIT and DEPOSIT NEXT keys.

Address

The memory address required by the START, READ IN, EXAMINE, and DEPOSIT keys is supplied by these switches. When one of these keys is depressed, the address given by the address switches is loaded into the MA (memory address) register and the key instruction is executed.

Register Group

Selects which of two groups of 12 registers to be selected by the register select switch. In the OFF position the normal registers may be viewed while the ON position allows viewing of the maintenance registers. (See Register Select Switch).

Clock

The ON position disables the real-time clock from issuing program interupt (PI) requests.

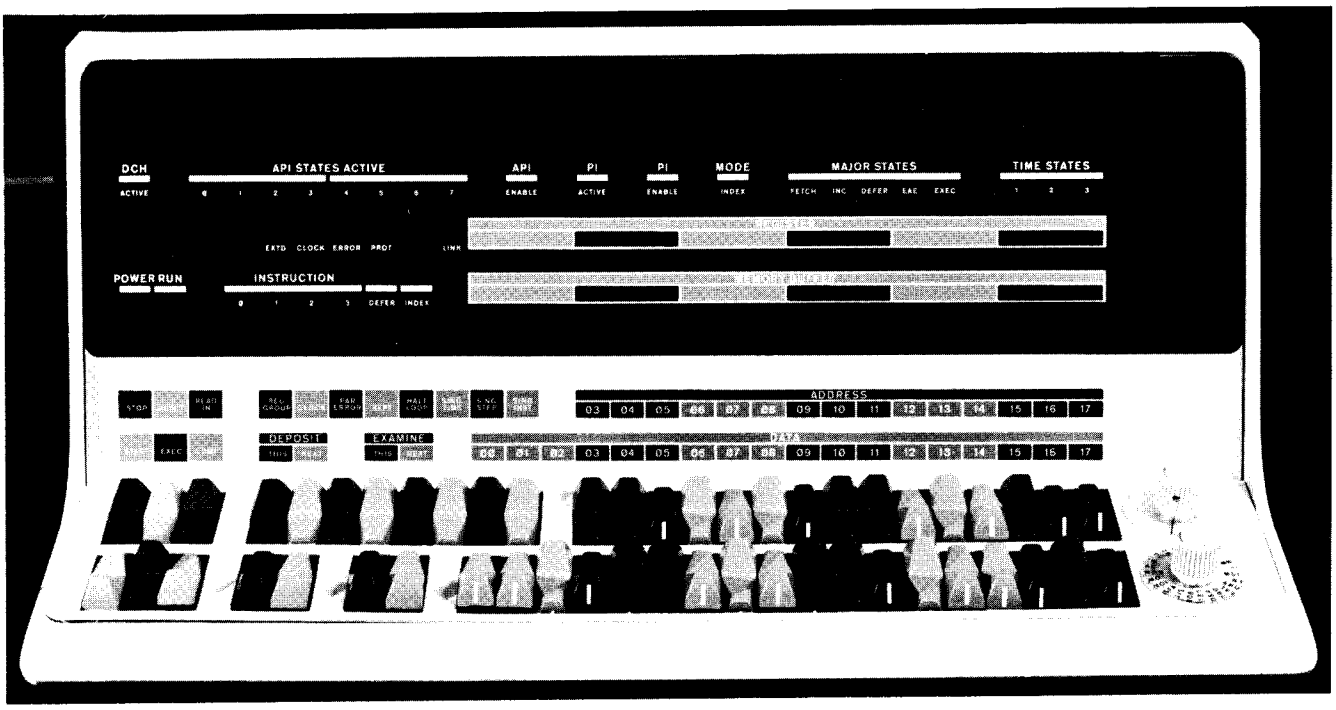


Figure 10-1. PDP-15 Console

Parity Error

The ON position disables parity error program interrupts from occurring.

Repeat

With this switch in the ON position, the processor will repeat the key function depressed by the operator at the rate specified by the repeat clock.

Start - program execution will restart at the repeat speed after the machine halts.

Execute - the instruction in the data switches will be executed at the repeat clock rate.

Continue - program execution will continue at the repeat speed after halting.

Deposit: This, Next, Examine: This, Next - the Deposit, Deposit Next or Examine Next function will be repeated.

Depressing STOP or turning off the repeat switch will halt the repeat action.

Halt Loop

Used in conjunction with single time, step or instruction. Enabling the halt loop switch and any one of the other three will cause the processor to continually repeat the time, step or instruction into which it has been advanced.

Single Time

Halts execution of the program after each time state has been completed.

Single Step

Halts execution of the program after each major state has been completed.

Single Instruction

Halts program execution after each instruction has been completed.

OPERATE CONTROL SWITCHES

The following momentary contact switches are on the console panel.

Number	Designation	Number	Designation
1	Start	1	Read In
1	Execute	1	Deposit This
1	Continue	1	Deposit Next
1	Stop	1	Examine This
1	Reset	1	Examine Next
		1	Protect

Start

Initiates program execution at the location specified by the address switches.

Execute

Executes the instruction in the data switches and halts after execution.

Continue

Resumes program execution from the location specified by the program counter (PC). Also used to advance machine while in single time, step or instruction.

Stop

Halts the processor after the completion of the present instruction.

Reset

Generates a power clear signal which clears all active registers and all control flip-flops except the program counter (PC).

Read-In

Initiates the read-in of paper tape punched in binary code (each set of three 6-bit lines read from tape forms one 18-bit computer word). Storage of words read-in begins at the memory location specified by the ADDRESS switches. At the completion of tape read-in, the processor reads the last word entered and executes it.

Deposit

Deposits the contents of the data switches into the memory location specified by the address switches. After the transfer, the memory locations address is in the OA (operand address) register and the contents of the accumulator switches are in the MO (memory out) register.

Deposit Next

Deposits the contents of the data switches into the location given by OA ±1. This permits the loading of sequential memory locations without the need of loading the address each time.

Examine

Places the contents of the memory location specified by the address switches into the memory buffer register. The address is loaded into the OA (operand address) register.

Examine Next

Places the contents of the memory location specified by the OA + 1 (operand address plus 1) into the memory buffer register. Sequential memory locations may be examined using the Examine-Next switch.

Protect

Initiates the memory protect feature. When utilized, the protect feature prevents the read-in of data into the specified "protected" memory locations.

SPECIAL SWITCHES

Register Select

The 12 position Register Select rotary switch can select the following registers for viewing in the REGISTER indicators (under control of the Register Group switch).

Register Group Switch OFF

AC	Accumulator
PC	Program Counter
OA	Operand Address
MQ	Multiplexer Quotient
L, SC	Priority Level/Step Counter
XR	Index Register
LR	Limit Register
EAE	EAE
DSR	Data Storage Register
IOB	I/O Bus
STA	I/O Status
MO	Memory Out

Register Group Switch ON

A BU	A Bus
B BU	B Bus
C BU	C Bus
SFT	Shift Bus
IOA	I/O Address
SUM	Sum Bus
M1	Maintenance I
M2	Maintenance II
MDL	Memory Data Lines
MA	Memory Address
MB	Memory Buffer
MST	Memory Status

Maintenance 1 and 2

Two positions of the Register Select switch M1 and M2, in conjunction with the Single Time switch, provide a visual display of active control and gating signal levels. For a given instruction, as the processor is stepped through the time states of each major state, the REGISTER indicators display a predictable sequence of enabling and strobing signals. The visual display relieves personnel of a large portion of the signal tracing normally associated with the maintenance of electronic systems. The sequences or maintenance "status words", are supplied as part of the system maintenance data.

Power/Repeat Rate

A variable-pot/switch provides the POWER ON/OFF control at one end of its rotation and variable repeat speed (approximately 1 Hz to 10 kHz) over the remainder of its rotation.

INDICATORS

The console indicator panel displays the following:

- Memory Buffer (18 bits)
- Register (18 bits)*
- Link
- Power
- Run
- Instruction Register
  - IR 0-3
  - Defer
  - Index
- Extended Enable
- Clock Enable
- Parity Error
- Protect
- DCH Active
- API States Active, 0-7
- API Enable
- PI Active
- PI Enable
- Index Mode
- Major States
  - Fetch
  - Increment
  - Defer
  - EAE
  - Execute
- Time States
  - 1
  - 2
  - 3

*The "register" indicators display the content of the register, bus or status word that is selected by the setting of the Register Select switch.

CHAPTER 11
SYSTEM SOFTWARE

PDP-15/10 COMPACT SOFTWARE SYSTEM

The PDP-15/10 COMPACT software system, which operates with the hardware configuration shown in Figure 11-1, is a concise programming system and includes a symbolic assembler, a text editor for creating programs on-line, debugging routines, utility routines, and mathematical routines. The system is designed to operate in the 4K or 8K paper-tape input/output environment of the basic PDP-15/10. Installations with 8192 words of core memory and high-speed paper tape with or without bulk storage may use the BASIC I/O MONITOR software to extend system capabilities.

Utility routines in the COMPACT software system include a Hardware Read-in Mode (HRM) punch routine, paper-tape handling routines, teletype I/O routines, an octal dump routine, and a memory scan routine used for scanning areas of memory for a particular bit configuration. For systems with DECtape, a FAST system can be used to retrieve frequently used programs.

COMPACT Assembler

A 2-pass system requiring less than 3072 words of core memory. The COMPACT Assembler has a useful set of selected pseudo ops for functions such as table formation, symbol table and variable control, and text handling.

COMPACT Debugging Routines

Debugging routines are included in the COMPACT Software system. ODT (Octal Debugging Technique) is a debugging aid that allows the user to conduct an interactive, on-line debugging session using octal numbers and teletype commands.

COMPACT Editor

Takes advantage of the powerful character string, search, and modification commands developed for the larger systems. It provides for the creation and/or identification of source programs and other ASCII text material by means of keyboard commands, and offers an efficient method for on-line processing of paper tapes.

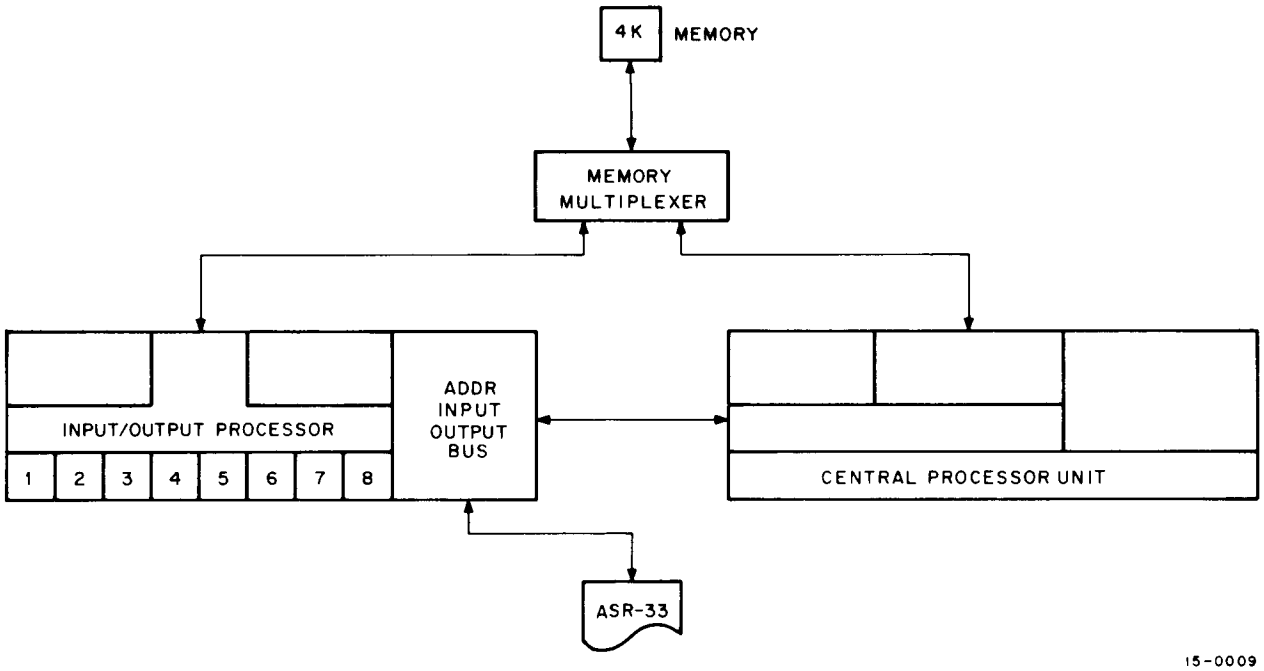


Figure 11-1. PDP-15/10

BASIC Monitor

This Monitor is available for configurations comprising 8192 words of core memory and high-speed paper tape reader/punch. It provides the link between the call for I/O, by either user or system programs, and the actual I/O execution. All input/output calls to system devices are serviced by Digital-supplied device handlers which reside in the input/output programming system (IOPS). The device handlers actually move the data between the program and the I/O devices. They are responsible for initialization of the device and for the performance of all other functions peculiar to a given I/O device, such as servicing of interrupts, in a real-time environment. User-supplied device handlers can be incorporated into the system to perform the above functions for special I/O devices.

PDP-15/20 ADVANCED MONITOR SYSTEM

The PDP-15/20 Advanced Monitor incorporates all the functions of the Basic I/O Monitor together with executive control of mass storage devices to provide fully automatic operation, including batch processing, keyboard interaction, and real-time control. The PDP-15/20 Advanced Monitor System has almost 50 basic commands that direct the operation of the hardware system (Figure 11-2). These commands perform three major functions:

a. Provide information about the system such as commands available and their functions; system configuration, error diagnostics, the standard logical-physical I/O device associations, I/O level programs available (device handlers), special memory registers and their functions.

b. Permit the standard physical-logic device associations to be modified; thereby enabling the dynamic allocation of devices at load-time. (This is a natural extension of device independence provided by the I/O monitor section of the Basic I/O Monitor.)

c. Supervise the loading and execution of all system and user programs, their associated I/O device handlers, and library subroutines, in addition to the generation of error messages and recovery procedures.

Coupled with keyboard control of system programs, the PDP-15/20 Advanced Monitor permits the user to deal with his entire problem (editing, assembling-compiling, loading, debugging and running) in a straightforward manner.

The PDP-15/20 Advanced Monitor (Figure 11-3) consists of a *systems loader, command decoder,* IOPS *routines, real-time clock handler, teletype handler,* an *error detector program,* and *device assignment tables* (DATS).

The *bootstrap loader* always resides in upper memory and is responsible for loading the Monitor into lower memory. Return calls from system or user programs cause restoration of control to the Monitor.

The Monitor *command decoder* recognizes requests for system programs and *loads* the system loader to bring in the requested program. In response to control cards or keyboard commands, it also manipulates the device assignment table to provide the device-independence. The Monitor *input/output system routines* (IOPS) include data-handling subroutines, device handlers, and interrupt service routines for the priority interrupt system as well as the teletype keyboard and printer. All other IOPS device handlers will be stored on the system device until required by object programs.

The Monitor also contains a *device assignment* for each table entry that may be used. Since the contents of the table can be altered by commands to the PDP-15/20 Advanced Monitor, actual I/O devices may be changed without altering the program references to these devices.

COMMON PDP-15 SOFTWARE

The following system software is supplied with all PDP-15 Systems.

FORTRAN IV - The PDP-15 FORTRAN IV compiler is a two-pass system which accepts statements written in the FORTRAN language and produces a relocatable object code capable

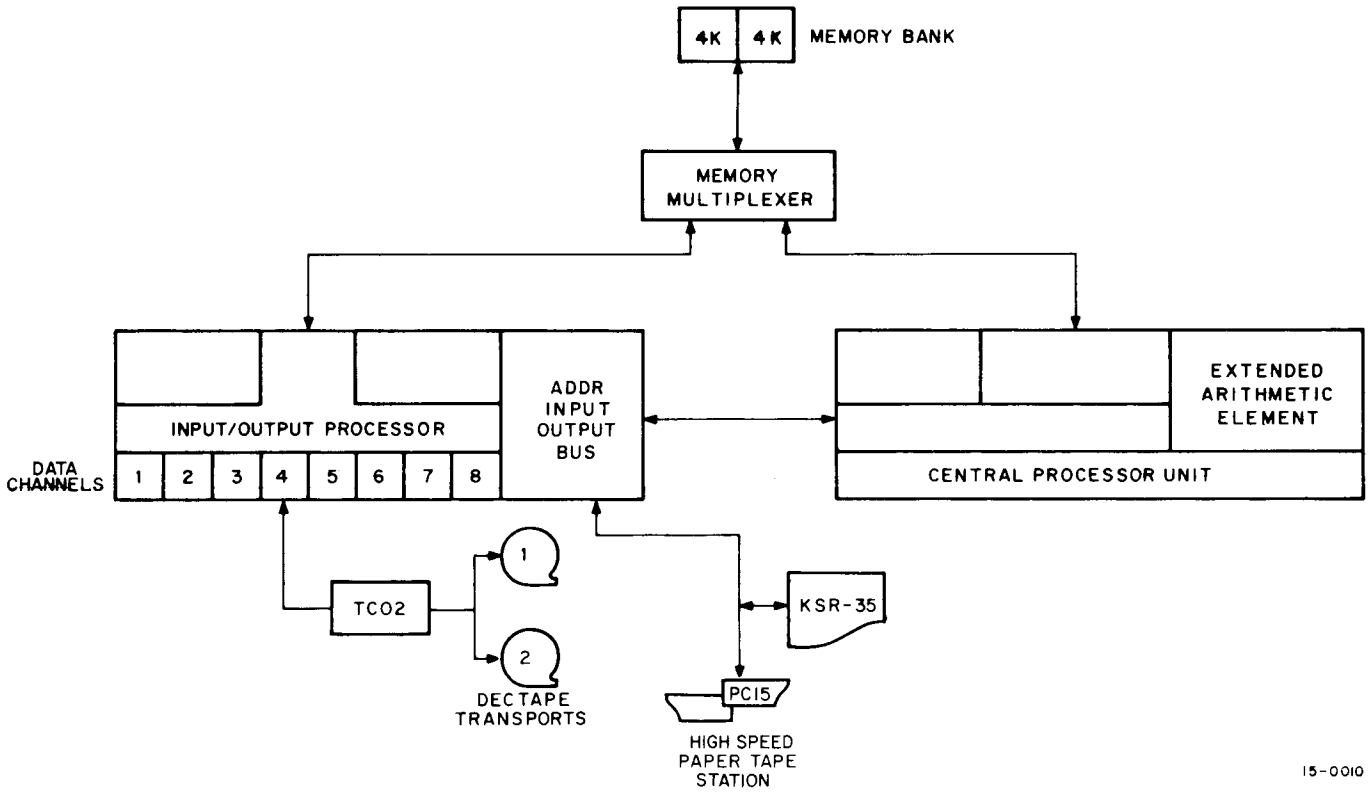


Figure 11-2. PDP-15/20

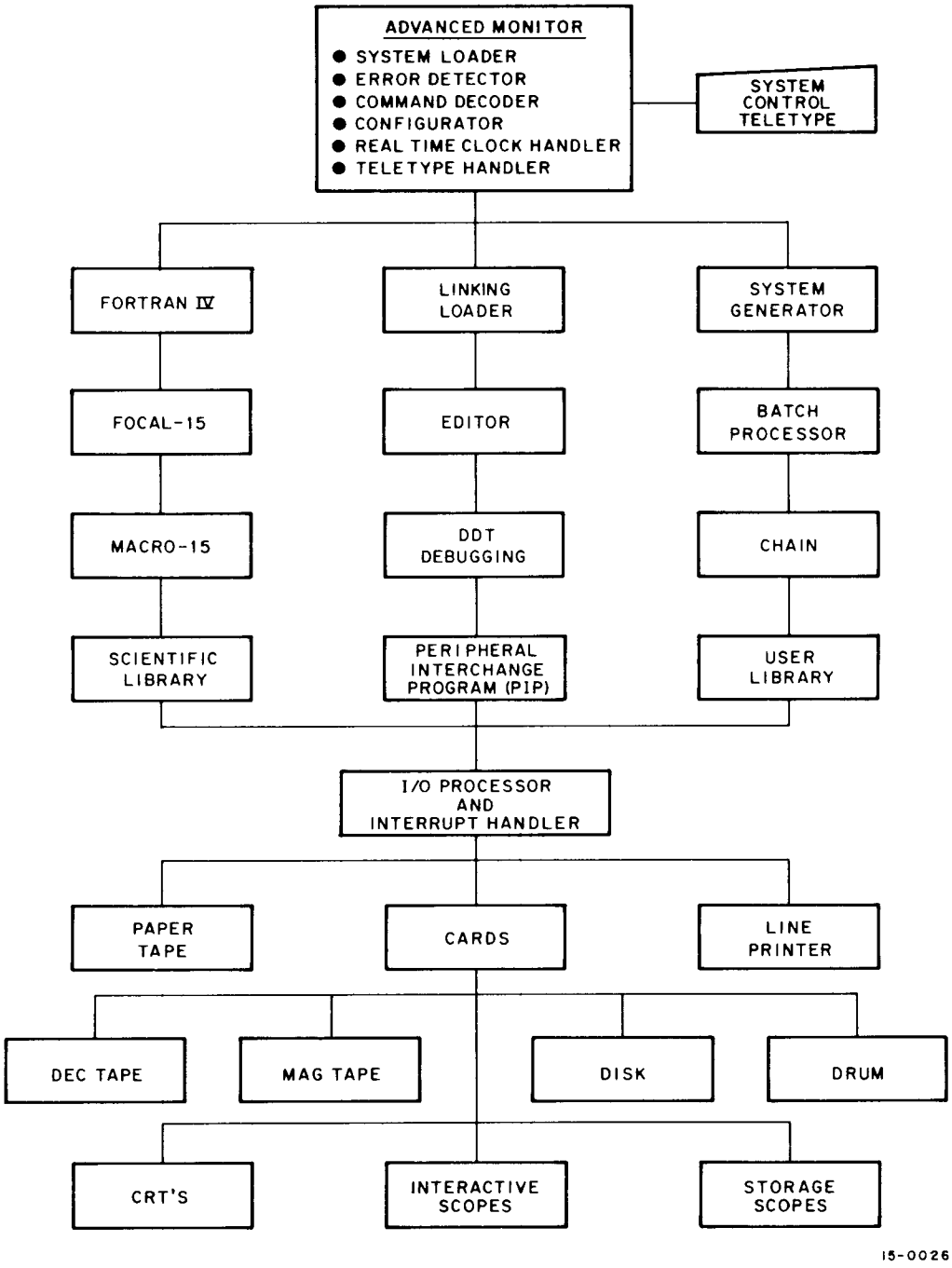


Figure 11-3. Advanced Monitor System

of being loaded by the Linking Loader program. It is compatible with USA FORTRAN IV, as defined in the USA Standard X3.9-1966, modified to allow the compiler to operate in 8,192 words of core storage.

This FORTRAN IV compiler operates with the PDP-15 program interrupt facility enabled and generates real-time programs that both operate with the program interrupt enabled and can work in conjunction with assembly language programs that recognize and service real-time devices. Subroutines written in either FORTRAN IV or the Macro Assembler language can be loaded with and called by FORTRAN IV main programs. Comprehensive source language diagnostics are produced during compilation, and a symbol table is generated for use in on-line debugging.

FOCAL - An on-line, interactive (conversational) algebraic language designed to help scientists, engineers, and students solve numerical problems. The language consists of short, easy-to-learn English imperative statements. Mathematical expressions are usually typed in standard notation. FOCAL puts the full calculating power and speed of the PDP-15 under easy conversational control. For example, FOCAL can be used for stimulating mathematical models, for curve plotting, for handling sets of simultaneous equations in n-dimensional arrays, and for solving many other kinds of problems.

Macro Assembler (MACRO-15) - Macro Assembler permits the programmer to use mnemonic symbols to represent operational codes, locations, and numeric data. The programmer can direct the assembler's processing by means of a full set of pseudo operations. An output listing can be obtained to show the programmer's source coding as well as the binary object code produced by the Assembler. PDP-15 users can also utilize highly sophisticated macro generating and calling facilities within the context of a symbolic assembler. Among the features of MACRO-15 are:

the ability to define and call nested macros;

conditional assembly based on the computational results of symbols or expressions;

repeat functions;

Boolean manipulation;

optional octal/symbolic listing;

two forms of radix control (octal and decimal) and two text modes (7-bit ASCII and 6-bit trimmed ASCII);

global symbols for easy linking of separately assembled programs;

choice of output format: relocatable, absolute binary (checksummed), or full binary (unchecksummed), capable of being loaded via the hardware READIN switch;

the ability to call I/O system macros which expand into IOPS calling sequences

Dynamic Debugging Technique (DDT-15) - A versatile tool for dynamic program checkout and modification. It allows an operator to load a program and run all, or selected portions, of it in a real-time interrupt environment under interactive supervision.

Control of DDT and program examination and modification are obtained via the teletype keyboard. A set of simple commands is available to allow the operator to insert a breakpoint, specify the number of program iterations before interrupting the program, and to start the program at any point. Other commands allow the operator to examine or alter any location symbolically and then rerun the program.

Text Editor - The PDP-15 ADVANCED Software System provides the ability to create or edit symbolic text utilizing any input or output device. The Editor operates on lines of ASCII text. A "context" method is employed throughout to identify the block of data which the user wishes to modify: that is, the block is specified by its ASCII text rather than by a numbering scheme imposed externally upon the text and not a part of it. Commands are available which facilitate insertion, deletion, and modification of data in the object file.

Peripheral Interchange Program (PIP-15) - Facilitates the manipulation and transfer of data files from any input device. It can be used to update

file descriptions, verify, delete, segment, or combine files, perform code conversions and copy tape.

Linking Loader - Loads in either relocatable or absolute format any PDP-15 FORTRAN IV or MACRO-15 object program. Among its tasks are loading and relocation of programs, loading of called subroutines, retrieval and loading of implied subroutines and IOPS routines, and loading and relocation of the necessary symbol tables.

Chain and Execute - Chaining in the PDP-15 ADVANCED Software System is a method of segmentation which allows for multiple core overlay of executable code and certain types of data areas. It reserves a portion of user core from one segment to another so that the user can communicate between segments. This core is reserved and used by the blank COMMON statement in FORTRAN IV and by the system pointers in MACRO-15. There are two system programs required for chaining.

a. Chain - a modified version of the Linking Loader which allows the user to build all the various segments (or chains) of his program into an executable file.

b. Execute - a control program which initiates loading of an executable file and transfers control from one segment (chain) to another.

The PDP-15/30 BACKGROUND/FOREGROUND Monitor System operating with the hardware configuration shown in Figure 11-4, is an extension of the Advanced Monitor system. Its open-ended design enables the system to handle a wide range of tasks, from high-speed data gathering applications such as those in physics to thousand-channel input/output applications such as warehouse inventory control. It allows the concurrent, time-shared use of the PDP-15/30 by protected foreground user programs with a background of batch processing, program development or low-priority user programs. With the BACKGROUND/FOREGROUND Monitor the user can:

a. Effectively have two processing systems - one for on-line data acquisition and control, one for off-line program development and data reduction - at the price of one system.

b. Achieve 100% utilization of his system, independent of data rates.

The foreground programs are assumed to be checked out and to operate from requests to the program interrupt or priority interrupt facilities. At load time, they have first priority over core memory and I/O devices, and at execution time, they have priority (according to their assigned priority levels) over processing time and their own I/O devices.

The background program (or sequential series of programs) is essentially the same as the single user program under the Advanced Monitor System; that is, it can be an assembly, a compilation, a debugging run, a production run, an editing task, or batch processing. It may use whatever facilities (core, I/O processing time, etc.) are available and are not required by the foreground programs.

The BACKGROUND/FOREGROUND Monitor oversees the time-shared use of the PDP-15/30 by the two co-resident programs and performs the following functions:

schedules processing time;

protects the foreground job's core;

protects the foreground job's I/O devices;

allows the sharing of multi-unit device handlers, such as DECtape, by both foreground and background jobs;

allows the queuing of jobs by priority through use of the software-initiated multi-level automatic priority interrupts (API);

allows the shared use of the system real-time clock to specified intervals;

allows communication between background and foreground jobs via core-to-core transfers.

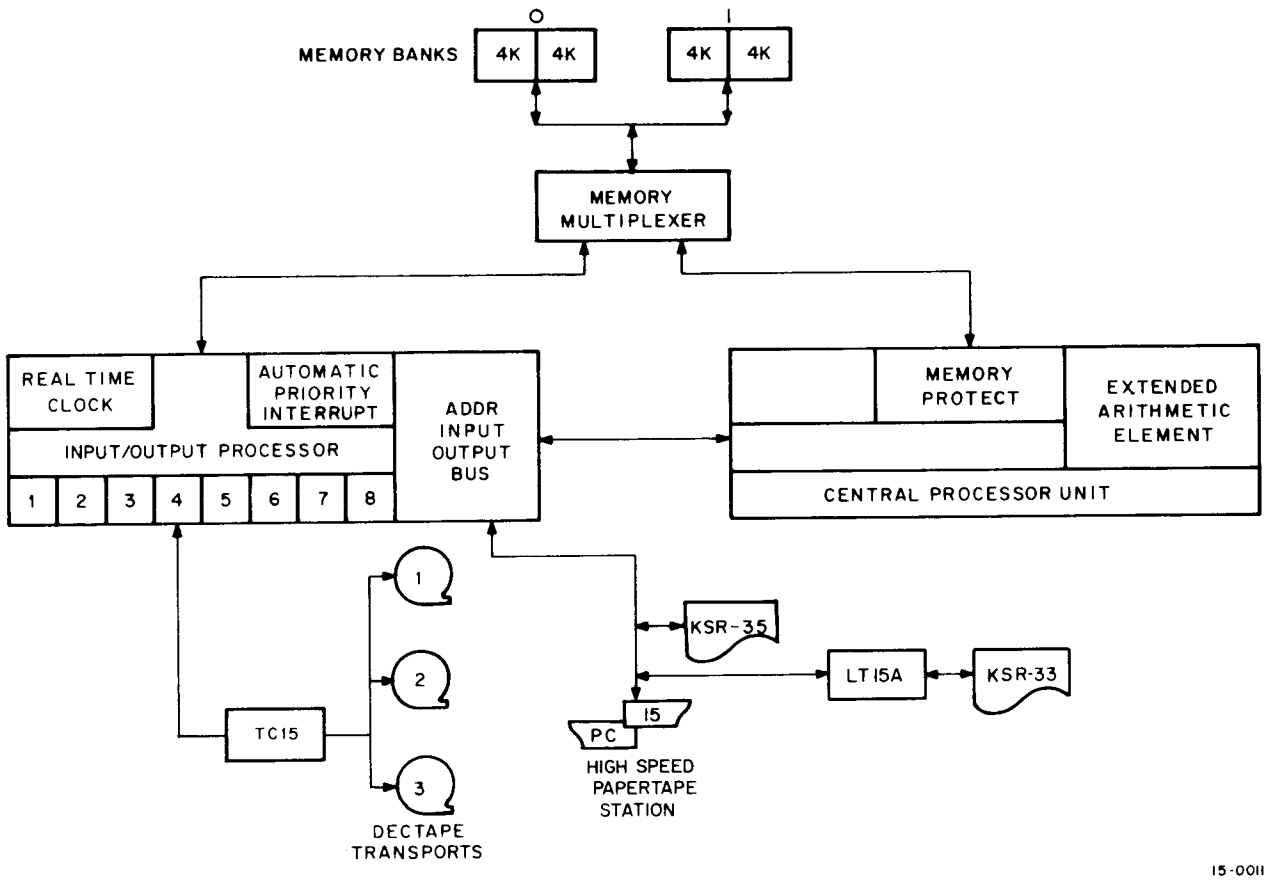


Figure 11-4. PDP-15/30 System

PDP-15/40 DISK-ORIENTED BACKGROUND/ FOREGROUND SYSTEM

The PDP-15/40 System (Figure 11-5) utilizes a disk version of the BACKGROUND/FOREGROUND Monitor. It contains all of the features described above in the PDP-15/30 BACKGROUND/FOREGROUND Monitor section. The disk system allows high-speed overlays, chaining, and system and user program loading. The limit of records that can be opened on the disk is limited only by available word space. The PDP-15/40 System contains 524,288 words of disk storage expandable up to 2,097,152 words.

ADDITIONAL SYSTEMS SOFTWARE

PDP-8-to-15 Translator

Used to translate programs written for the PDP-8 in PAL-III, PAL-D, or MACRO-8 assembly language to MACRO-15 assembly language, so that they can be assembled and executed within the PDP-15 ADVANCED Software environment. The purpose of the Translator is not to produce a program which runs on the PDP-15 by simulating the PDP-8, but rather to do the straightforward portion of the translation and clearly indicate to the programmer those parts of the code which require review in the light of the PDP-15's greater word length and more power instruction set.

STATPAC

A comprehensive and open-ended package of modular statistical programs designed to operate under the PDP-15/20 ADVANCED Monitor, is an easy way for a user with limited computer knowledge to obtain statistically meaningful results from data. STATPAC includes modules for

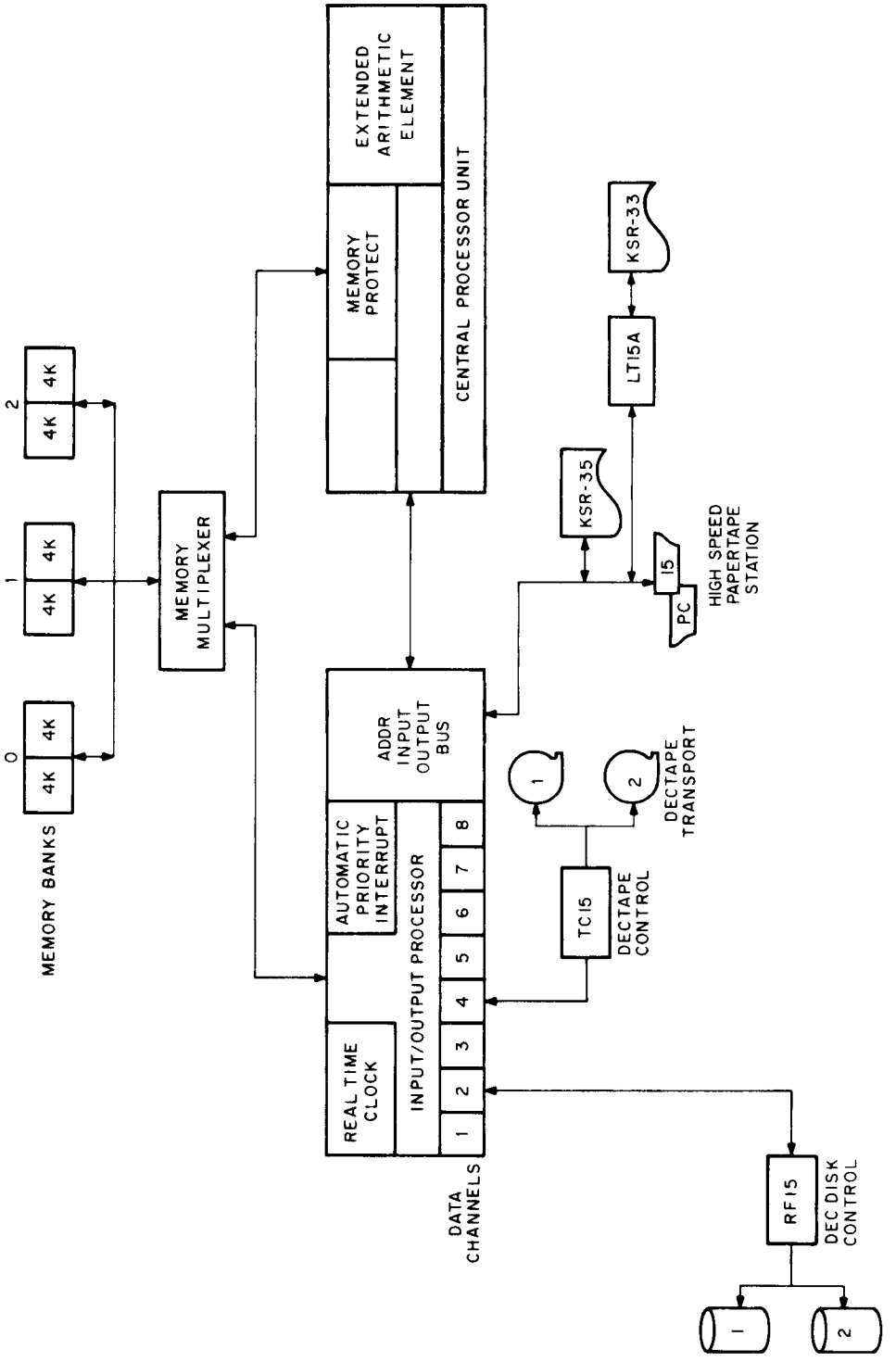


Figure 11-5. PDP-15/40 System

control, input, descriptive statistics, stepwise linear regression, and multiple linear regression functions.

DECUS

The Digital Equipment Computer Users Society, encourages the exchange of programs and ideas among its several thousand members. Semi-annual meetings provide a forum for papers, while the program library allows the exchange of programs of common interest.

DIAGNOSTICS

MAINDEC Diagnostic Programs

MAINDEC Diagnostic programs are provided for locating hardware malfunctions within the processor, memory, and I/O equipment. They run under a systems exercisor. Simple teletype commands load and execute requested diagnostics. Several diagnostics may be executed concurrently to maximize system's interaction.

The diagnostic programs are designed to make troubleshooting fast and straightforward by selectively exercising every circuit in the machine. Instructions and procedures for loading, operating, and interpreting the results of diagnostic tests are written in clear, simple language, so that beginning maintenance technicians can use them easily.

Detailed error messages are printed out to tell the technician exactly which instruction or bit configuration, has failed. Error codes help direct the troubleshooter to specific modules when a fault condition is detected.

Among the MAINDEC diagnostics are: The Basic Processor Test and the Extended Processor Test. The Basic Test incrementally checks the entire instruction repertoire, performing 1500 unique tests, and in each case, halts with specific instructions for the troubleshooter. If the Basic Test fails to detect the trouble, the Extended Test uses random number techniques to test the logic for many combinations of data manipulation and addressing problems, runs memory test patterns, performs system tests on I/O devices and controls, and many other tests.

A valuable tool for check-out and troubleshooting, MAINDEC diagnostics contribute to the high productivity of the PDP-15 by minimizing down time.

APPENDIX A
INSTALLATION PLANNING

PHYSICAL CONFIGURATION

The basic PDP-15 is housed in a standard 19-in. cabinet (Figures A-1 and A-3 with overall dimensions of 21-11/16 in. wide, 30-in. deep and 71-7/16 in. high. The PDP-15 is painted black with grey end panels and a two-tone blue console. The console can be purchased in three versions: as a flush mounted console which is table mounted, as a tilted console with a table, or as a remote console. The table projects forward 19-5/16 in. and a rear clearance of 18-1/2 in. is needed for access to the logic.

All of the standard PDP-15 System logic is housed in a steal enclosure with cooling fans. Each cabinet uses a large fan which pulls filtered air in from the top of the cabinet - keeping the complete cabinet under pressure.

In the basic PDP-15 cabinet, the console, power supply, API, parity, memory protect, central processor, I/O processor, EAE, real-time clock, power fail and the first 8192 words of memory are all mounted in the front portion of the cabinet. Additional memory to 32,768 words is mounted on the back door.

Other options are added to the PDP-15 by attaching 19-in. cabinets to either side of the basic System, according to the configuration shown in Figure A-4. Connections to these options are made from the PDP-15 I/O Processor via I/O bus cables.

Several options including disks, displays, industry-compatible tape transports, the card reader, line printers and plotters are composed, in part, of free standing units.

PLACEMENT OF OPTIONS

Cabinets are numbered 1 through n with the numbers always running from left to right. All cabinets are standard DEC 19-in. type weighting 300 lbs. with net capacities of 500 lbs. recommended and 800 lbs. maximum. Each cabinet can hold eleven standard 5-1/2-in. mounting panels of logic plus a 5-1/2-in. indicator panel at the top. Figure A-4 shows the placement of options and peripherals.

Figures A-5 through A-8 show the configuration of the PDP-15/10, 20, 30, and 40, respectively.

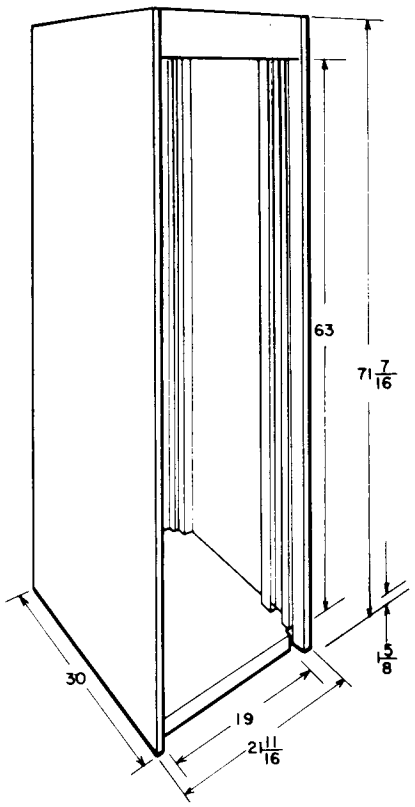


Figure A-1. Basic PDP-15 Cabinet

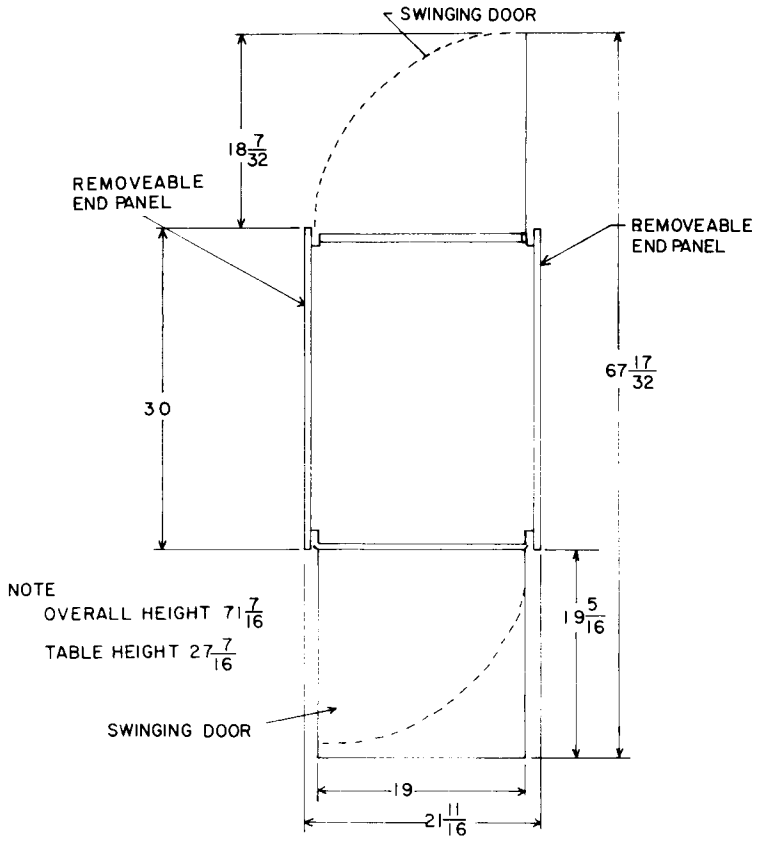


Figure A-2. Top View of Basic PDP-15 Cabinet

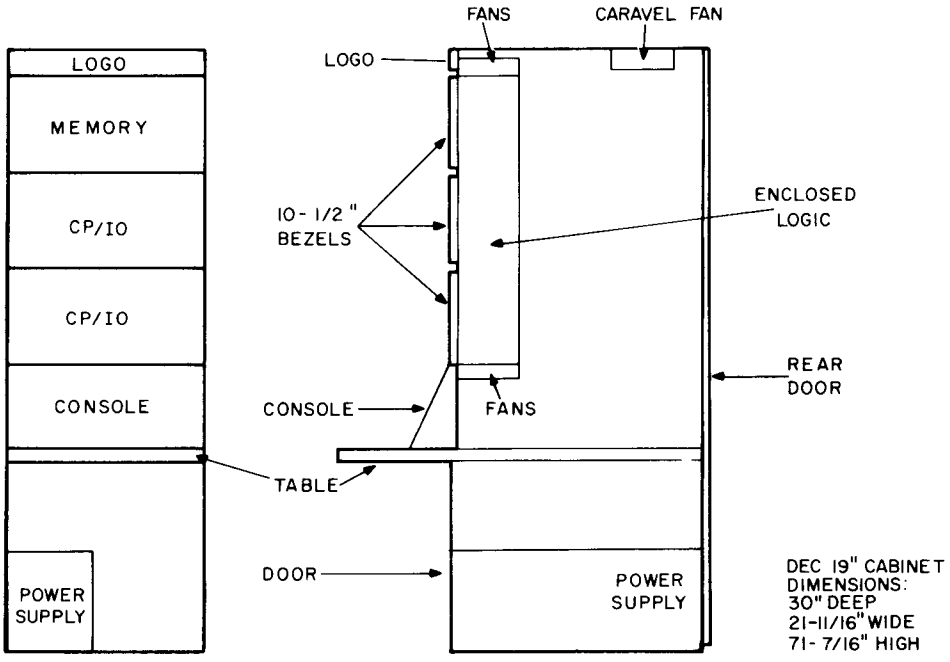
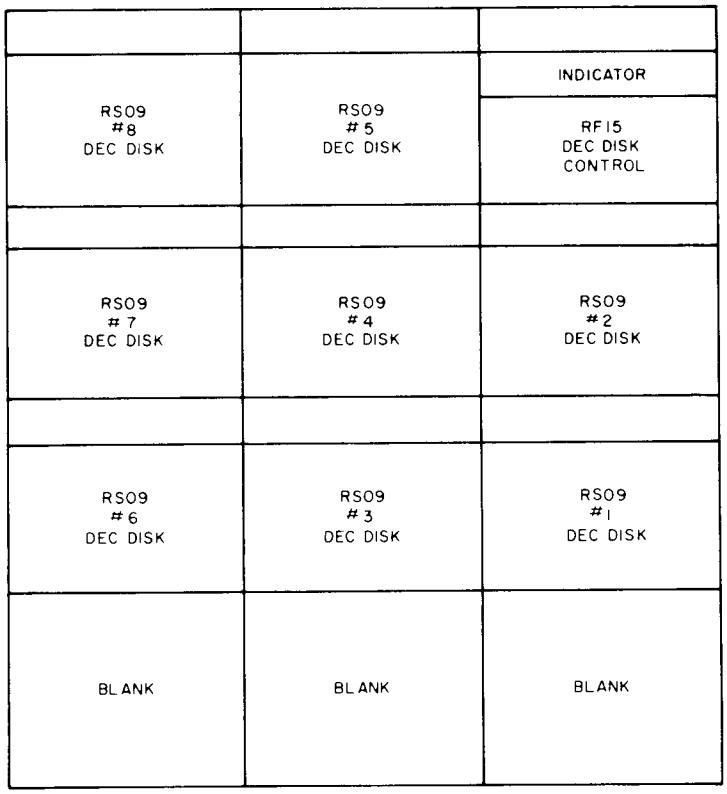


Figure A-3. PDP-15 Basic Configuration



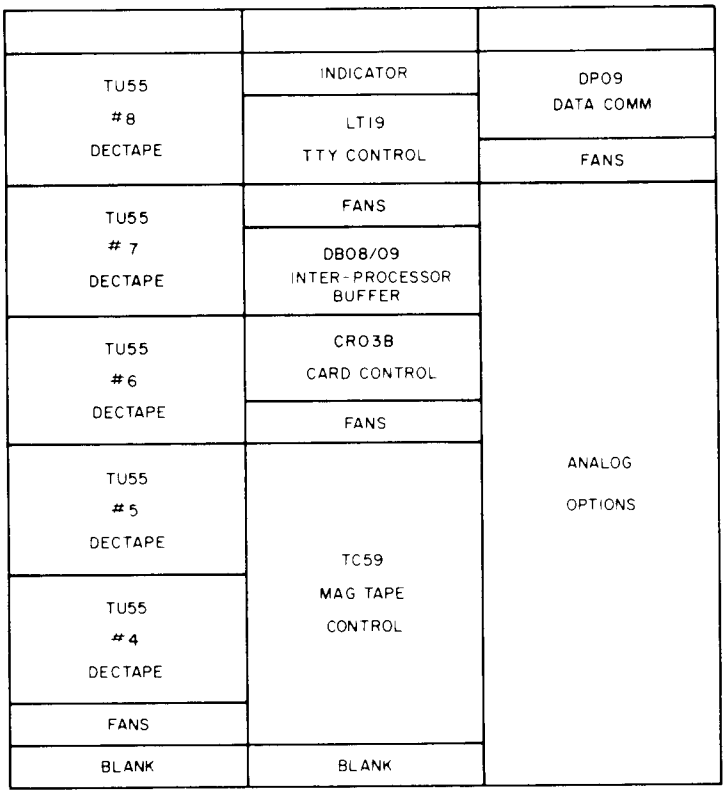


Figure A-4. Option and Peripheral Physical Configuration (Sheet 1)

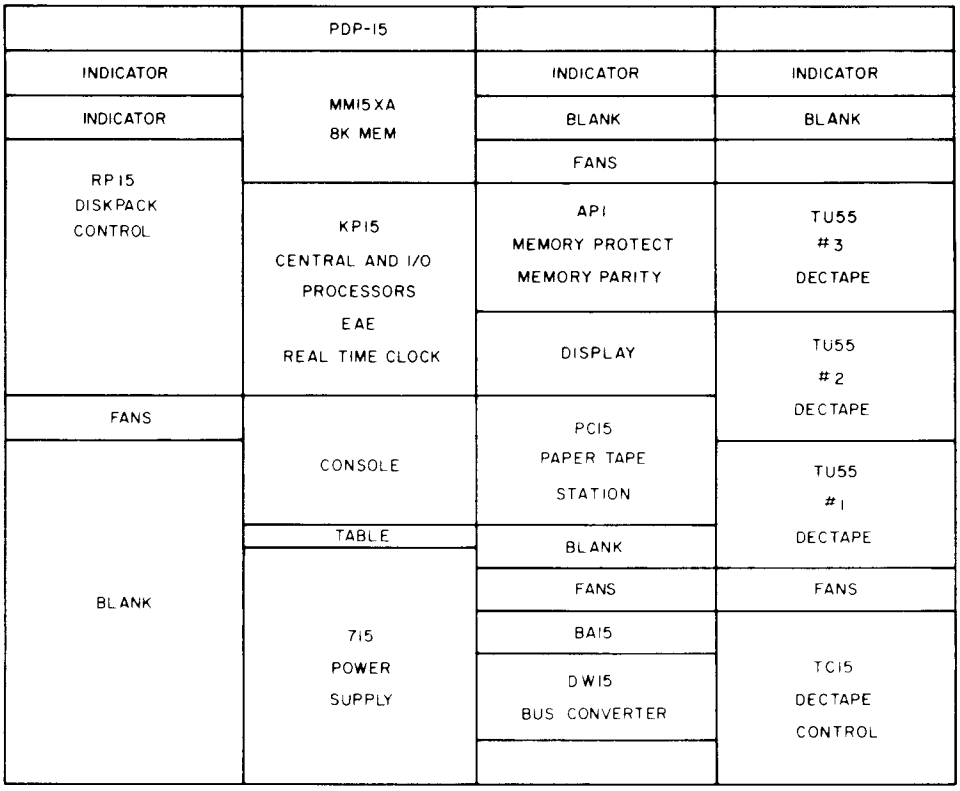


Figure A-4. Option and Peripheral Physical Configuration (Sheet 2)

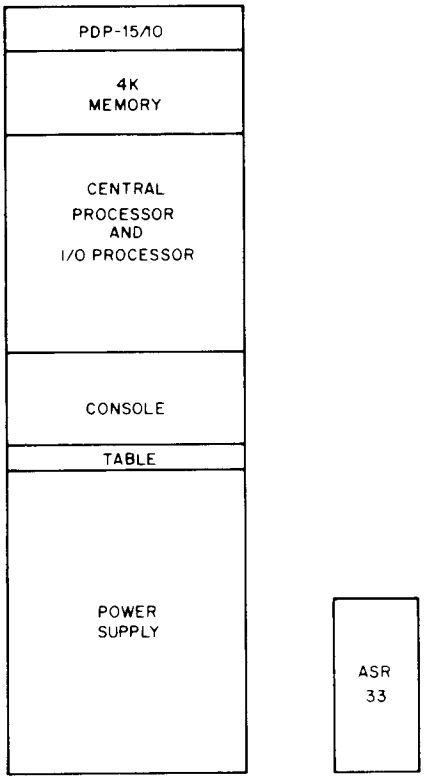


Figure A-5. PDP-15/10

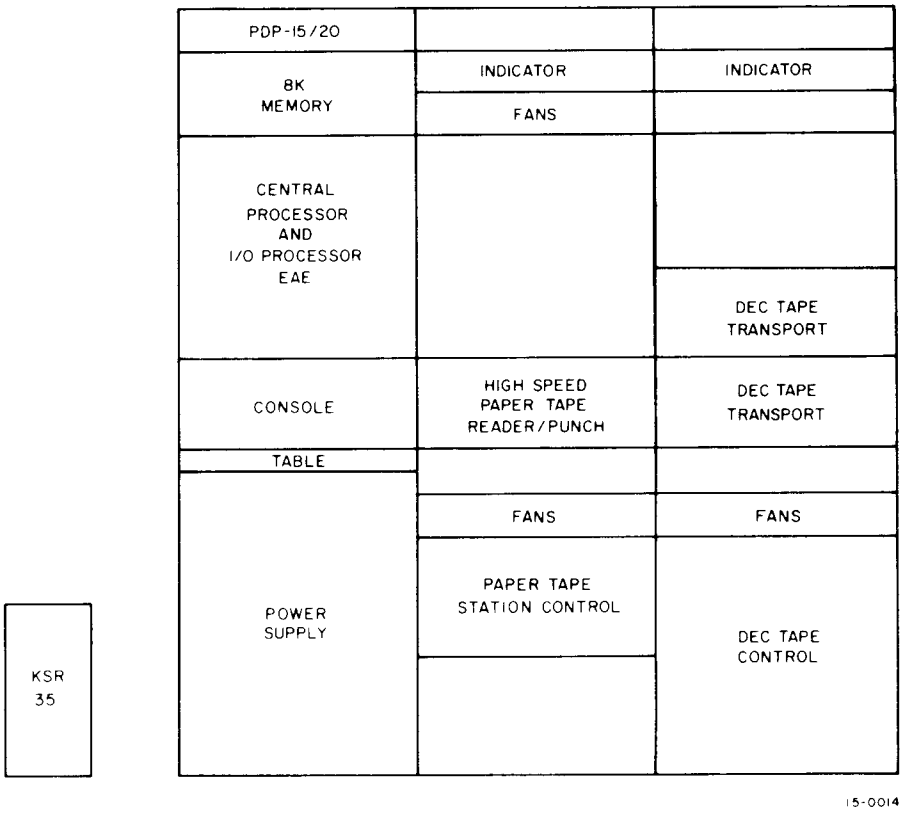


Figure A-6. PDP-15/20

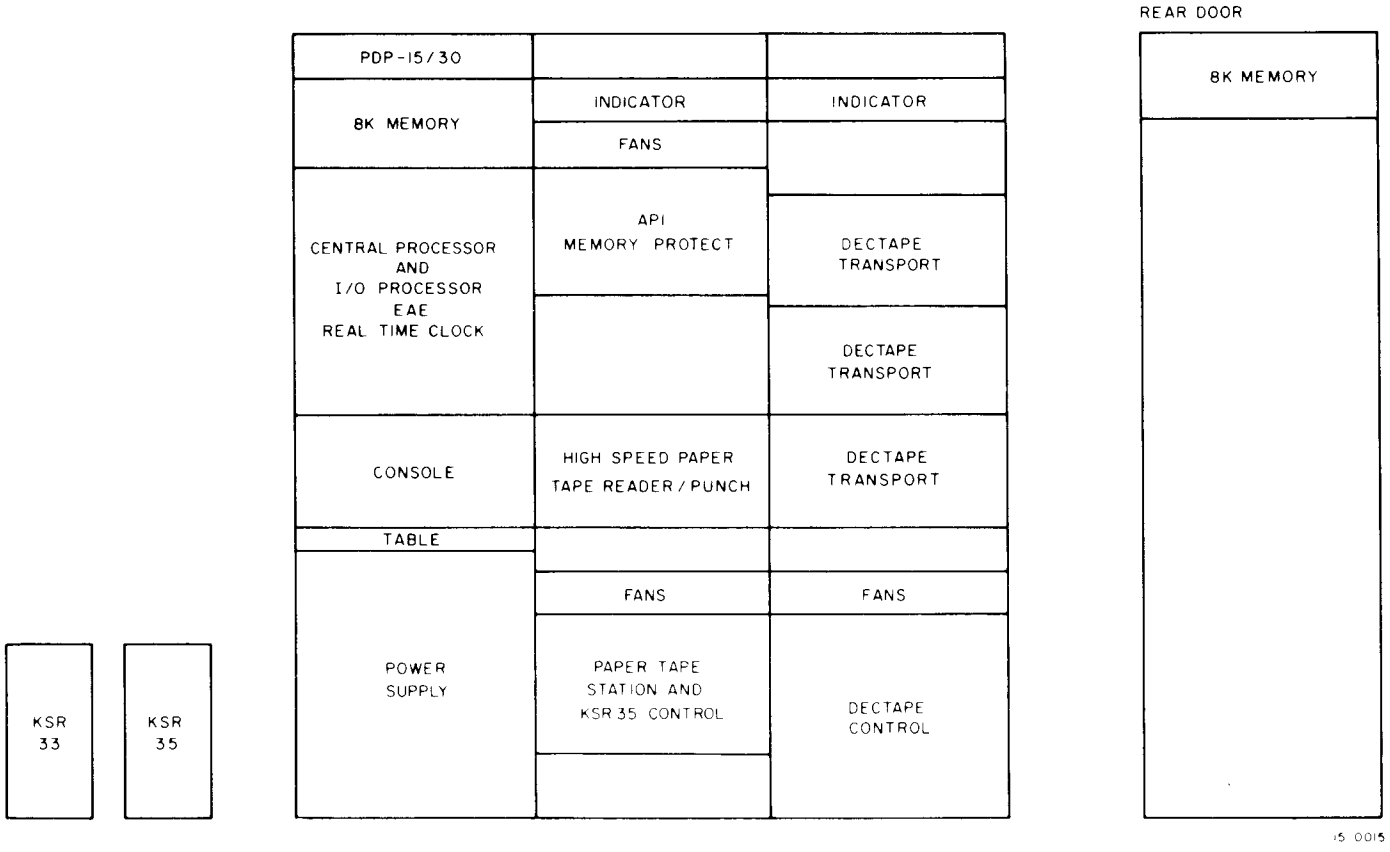


Figure A-7. PDP-15/30

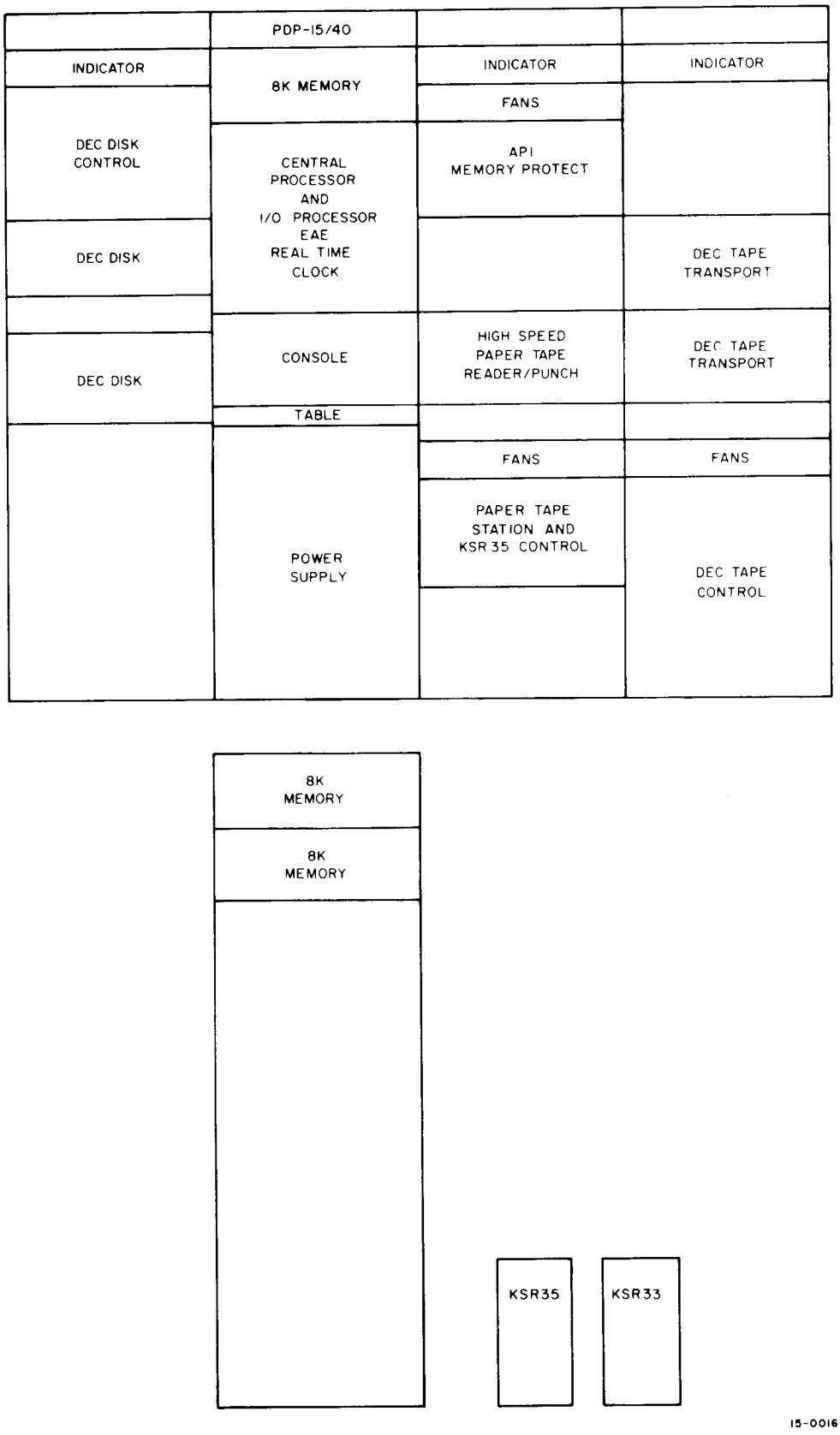


Figure A-8. PDP-15/40

Table A-1
PDP-15, Extra Memory, Free-Standing Options and Their Controls

Name	Cabinet Dimensions			Options Required	Service Clearance			Height of Interface (19 in. logic)	AC Current		Heat Dissipation (btu/hr)	Power Dissipation (kw)	Cable Length	Comments
	Height (in.)	Width (in.)	Depth (in.)		Weight (lb)	Front (in.)	Rear (in.)		Nominal (amps)	Surge (amps)				
Standard PDP-15	71-7/16	21-11/16	30	--			20	--						
Extra Memory, Type MM15 A/B/C														Rear door of basic cabinet can hold up to three 8K memory banks
Magnetic Tape Transports, Type TU20 and TU20A	69-1/8	22-1/4	27-1/16	TC59	400	19	19	See TC59D	8	12	2300	0.62	10 ft	60°F to 80°F 40% to 60%
200 cpm Card Reader, Type CR03B	50	30	17	DW15A	200	--	6-5/8	--	1.3	7.0	495	0.15	10 ft	
Incremental Plotter,								See 350						
Calcomp Model 563	9-3/4	39-3/8	14-3/4	XY15	53	--	--	Control	1.12	2	425	0.125	10 ft	
Calcomp Model 565	9-3/4	18	14-3/4	XY15	53	--	--		1.5	3	580	0.17	10 ft	
Data Communications System, Type 680	(See 680 Handbook)			DB98A	--	--	--	See DB98A	--	--	--	--	12 ft	

Table A-2
Hardware and Logic Options for 19-Inch Cabinets

Name	Logic Height (in.)	Number of Mounting Panels	Options Required	Approx Weight (lb)	AC Current (amps) Nominal	AC Current (amps) Surge	Heat Dissipation (btu/hr)	Power Dissipation (kw)	Comments
IPB, Type DB99A	10-1/2	2	–		0.39	0.72	133	0.042	
IPB, Type DB98A									
PDP-15 or PDP-9 End	10-1/2	2	–		0.39	0.72	133	0.042	
PDP-8 End	10-1/2	2	–		0.31	0.6	111	0.032	
DECtape Control, Type TC15	15-3/4	3	–	38	0.60	1.0	207	0.058	
DECtape Transport, Type TU55	10-1/2	2	TC02	35	0.5	3.0	410	0.170	
Magnetic Tape Control Type TC59D	21	4	–	50	2.0	5.0	1000	0.23	
Incremental Plotter Control, Type XY15	10-1/2	2	–	25					
Multistation Teletype Control, Type LT19A	10-1/2	2	–	25			47	0.014	
Line Unit, Type LT19B	–	–	LT19A		0.13	0.24	–	–	
Tektronix Scope, RM503 or RM564	7	2	VP15						
ADC/Multiplexer, Type AF01B	8-3/4	2	–	50	0.5		189	0.055	
IDVM/Scanner, Type AF04B									
200 Channels	24	Separate	–	220	4.0	8.8	1564	0.460	Weights include cabinet
400 Channels	34	Cabinet	–	240	4.5	9.8	1756	0.517	
600 Channels	44		–	260	5.0	11	1950	0.575	
800 Channels	54	Separate	–	280	5.5	11.9	2100	0.62	
1000 Channels	64	Cabinet	–	300	6.0	13.2	2350	0.69	
DAC Controller, Type AA05B	8-3/4	2	–	60	0.5	1.1	306	0.09	
DAC Controller Extension, Type AA07B	8-3/4	2	AA05B	30	0.5	1.1	306	0.09	

Digital Equipment Corporation
Maynard, Massachusetts

digital

printed in U.S.A.