PDP-11 UNIBUS Processor Handbook

PDP-11/84, PDP-11/44, and PDP-11/24

digital

digital
PDP-11 UNIBUS Processor Handbook

PDP-11 UNIBUS Processor Handbook

PDP-11/84, PDP-11/44, and PDP-11/24

digital

Contents

Chapter 3 ▪ PDP-11/44

Chapter 4 • PDP-11/24

Chapter 5 • UNIBUS Technical Description

Preface

This handbook is a reference guide that focuses on Digital's PDP-11 UNIBUS processors and details the UNIBUS specifications. Written with both evaluators and users in mind, this handbook will help you choose the processor that best meets the needs of your application.

Chapters 1 through 4 supply a discussion of the key functions of the following PDP-11 UNIBUS processors.

- PDP-11/84
- PDP-11/44
- PDP-11/24

Chapter 5 describes the technical specifications of the PDP-11 UNIBUS to assist in configuring large, complex PDP-11 systems and to aid in the design of interfaces for PDP-11s.

After reading this handbook, you may wish to learn more about PDP-11 architecture. The *PDP-11 Architecture Handbook* is a companion to this handbook and describes the key elements of the PDP-11 architecture, including addressing modes, the PDP-11 instruction set, memory management, PDP-11 floating point, commercial instruction set, and the assignment of addresses and vectors. This and other handbooks that you can order are listed in the documentation section of Chapter 1.

A postage-paid Reader's Comments card is located at the back of this handbook. We hope you will take the time to fill it out. Your suggestions will help us to continue to present literature that meets your needs.

The UNIBUS, a single, bidirectional, asynchronous bus, is the connecting device that makes possible the strengths and flexibilities of the PDP-11 family members discussed in this book. The UNIBUS is a high-speed communications path that links system components and peripheral devices and provides system-to-system connectability. This unique data bus provides the hardware and software backbone of the PDP-11/84, PDP-11/44, and PDP-11/24 processors.

The UNIBUS was the first data bus in the history of the minicomputer industry to enable devices to send, receive, or exchange data without processor intervention and without intermediate buffering in memory. PDP-11 architecture takes advantage of the UNIBUS in its method of addressing peripheral devices. Memory elements, such as main memory or any read-only or solid-state memories, have ascending addresses starting at zero, while registers that store input/output (I/O) data or the status of individual peripheral devices have addresses in the highest 8 Kbytes of addressing space. Communications between any two devices on the bus use a master/slave relationship and priority arbitration. Details of this unique UNIBUS design are highlighted in chapter 5 of this handbook.

Each member of the PDP-11 family of computers shares a common architecture. Each is based on a 16-bit wordlength, a common instruction set, and the same addressing techniques. Each also shares the same data management utilities, the same I/O systems, and the same programming languages. The software written for one member of the PDP-11 family runs on other PDP-11 family members. The peripherals and I/O systems for the UNIBUS PDP-11 family are also compatible. In other words, the success of this family of computers is a function of their common architecture that provides compatibility across all models.

Digital has provided customers with a clear growth path to expand their computer systems as their requirements demand. You can upgrade or extend any UNIBUS system by adding memory and peripherals without worrying about major incompatibilities in your computing system.

Beyond the architecture and software commonalities, the processors described in this handbook are bound together by the UNIBUS. Yet each PDP-11 processor has features and capabilities uniquely suited for various applications. Some functionally similar features have been accomplished with different implementations. Therefore, there is some repetition of information in the chapters describing the individual processor members. It is often necessary to discuss each separately because what may appear to be very subtle differences in operations may actually be crucial to a certain processor's uniqueness.

Table 1-1 lists many of the basic UNIBUS system functions and five of the UNIBUS PDP-11 processors for comparison. This chart allows you to compare the processors quickly, especially in relation to the latest member, the PDP-11/84. The X denotes the processor is equipped with that particular function.

• High-performance PDP-11/84 System

The year 1985 marks the 15th anniversary of the PDP-11 UNIBUS computer, and also the advent of Digital's new, high-speed minicomputer—the PDP-11/84. It is the most powerful, yet most cost-effective, UNIBUS PDP-11 computer ever designed. Based on the J-11 chipset with cache memory, the PDP-11/84 processor surpasses PDP-11/70 performance, yet fits into the price range of the current PDP-11 processors. The introduction of the PDP-11/84 establishes the fifth generation of UNIBUS PDP-11 computers. Figure 1-1 represents the enhancement of performance/functionality versus price with the development of each succeeding PDP-11 generation. The PDP-11/84 clearly provides the price/performance advantage over previous PDP-11 designs.

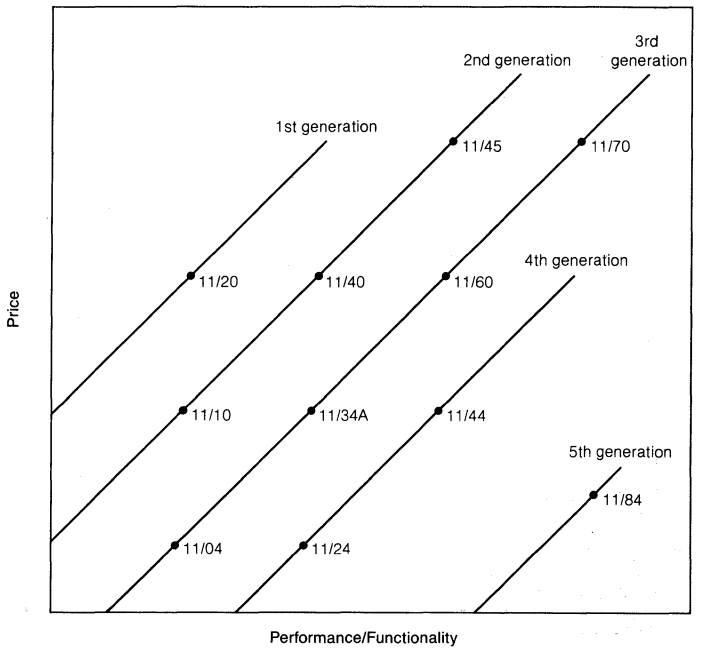


Figure 1-1 • UNIBUS PDP-11 Performance/Functionality versus Price

Table 1-1 ▪ Functionality Matrix

Function	PDP-11/84	PDP-11/44
Basic PDP-11 Instruction Set with Extended Instruction Set (multiply, divide, longshift)	X	X
Floating-point Processor	Microcode standard Hardware standard	Hardware optional
Commercial Instruction Set (CIS)		Optional
Memory-management Unit	X	X
22-bit Physical Addressing	X	X
Maximum Memory	3,840 Kbytes	3,840 Kbytes
Console	ASCII	ASCII
Battery Backup	Optional	Optional
Relative Processing Power	1.1	.75
Expansion Slots	7 hex/quad and 1 quad	1 hex/quad and 1 quad
Open System Units	0 in 10.5-inch box 2 in cabinet	3 in 10.5-inch box
Onboard Serial-line Units	1	2
Line-time Clock	X	X
Kernel and User Modes	X	X
Supervisor Mode	X	X
Separation of Instructions and Data	X	X
Dual-register Set	X	
Packaging		
Box	10.5-inch	10.5-inch
Cabinet	42.0-inch	10.5-inch box within 42.0-inch cabinet

PDP-11/24	PDP-11/34A	PDP-11/70
X	X	X
Microcode optional Hardware optional	Hardware optional	Hardware optional
Optional		
X	X	X
Optional		X
248 Kbytes 3,840 Kbytes optional	248 Kbytes	3,840 Kbytes
ASCII	ASCII standard Hardware optional	Hardware standard ASCII optional*
Optional	Optional	Optional
.30	.50	1.0
5 hex/quad and 1 quad	3 hex/quad and 1 quad	3 hex/quad and 1 quad
0 in 5.25-inch box 4 in 10.5-inch box	0 in 5.25-inch box 3 in 10.5-inch box	0 in cabinet
2	1	1
X	X	X
X	X	X
		X
		X
		X
5.25-inch 10.5-inch 10.5-inch box within 42.0-inch cabinet	5.25-inch 10.5-inch 10.5-inch box within 42.0-inch cabinet	60.0-inch or 72.0 inch

* Provided by Digital's Field Service.

The powerful J-11 chipset is an example of Digital's sophisticated technology, providing increased performance in a smaller package. The chipset itself performs functions that once required many separate boards. Based on CMOS technology, the J-11 microprocessor has 16-bit I/O and a 32-bit internal data path. It can address up to 4 Mbytes of memory. This 60-pin package implements the full PDP-11 instruction set including hardware multiply/divide, floating point instructions, and online debugging technique (ODT). In addition, the chipset contains memory management logic and cache-control registers. It also handles all data and address transfers.

▪ Software

The large installed base of PDP-11 computers with their associated software means that software for your application will be easy to find. You can choose from proven software developed and supported by Digital and you can choose from software supported by third parties. An important source of software is DECUS, the Digital Equipment Computer Users Society. DECUS is one of the largest and most active user groups in the computer industry and membership is available to owners of Digital's computers.

The PDP-11 UNIBUS systems detailed in this handbook offer simple, easy programming with a rich array of languages and software tools. Thousands of application software packages already exist to serve virtually every area of business and industry. The same software that runs on Digital's smaller, low-priced PDP-11 systems will run on our larger models. A directory of over 2000 available software offerings for PDP-11s is listed in Digital's two-volume *PDP-11 Software Source Book*.

The Digital PDP-11 16-bit operating systems meet realtime, timesharing, and batch proccessing demands efficiently and effectively in a host of environments—from small, dedicated laboratory and industrial control systems to large, multiuser information management systems. For more information about PDP-11 operating systems and layered products, see the *PDP-11 Software Handbook*. It includes a general description of each operating system, the file structure and data-handling facilities, the user interfaces, the system utilities, and the languages supported by each processor.

PDP-11 UNIBUS processors support multiple operating systems so that the right hardware, operating system, and application software can be combined to meet your exact requirements. Table 1-2 gives a short description of the PDP-11 operating systems and the availability of each operating system on the PDP-11 processors.

Table 1-2 • PDP-11 Operating Systems

Name	Description	Processor
RT-11	A single-user, multi-job system, ideal for dedicated realtime or small business applications	PDP-11/84, 11/44, 11/34A, 11/24, 11/70
CTS-300	A multiuser, commercial timesharing system that combines DIBOL with RT-11	PDP-11/84, 11/44, 11/34A, 11/24, 11/70
DSM-11	A timesharing system oriented toward large database applications	PDP-11/84, 11/44, 11/24, 11/70
RSTS/E	A general-purpose, time-sharing system that provides a fast response time in multiuser applications and program development	PDP-11/84, 11/44, 11/34A, 11/24, 11/70
RSX-11M-PLUS	Multiuser, multiprogramming realtime systems that support multiple languages, program development tools, and a variety of utilities	PDP-11/84, 11/44, 11/24, 11/70
RSX-11M RSX-11S	Subset of RSX-11M-PLUS appropriate for small-memory PDP-11s. RSX-11S is a memory-only system	PDP-11/84, 11/44, 11/34A, 11/24, 11/70
IAS	Interactive Application System allows realtime application execution concurrent with timeshared processing	PDP-11/84, 11/44, 11/24, 11/70
ULTRIX-11	Digital's enhanced native-mode UNIX software that is upward compatible with Digital's ULTRIX-32 operating system	PDP-11/84, 11/44, 11/24, 11/70

▪ Networks

Digital offers extensive capabilities that permit the linking of computers and terminals into flexible configurations called networks. Networking lets computer systems amd terminals, whether located within a facility or around the world, share resources and exchange information, files, and programs. The smaller computers in a network have access to the powerful capabilities of larger systems, while the larger computers can take advantage of smaller dedicated systems chosen for specific application environments.

The PDP-11 UNIBUS processors support a wide range of users while simultaneously connecting them all to a network of systems. Digital's powerful communications software, DECnet, enables two or more Digital computer systems —16-bit PDP-11s, 32-bit VAXes, and 36-bit DECsystem-10s and DECSYSTEM-20s —to form a network. DECnet offers task-to-task communications, remote file access, utilities for network file transfer, heterogeneous network command terminal support, and network resource sharing.

The Ethernet-to-UNIBUS high-performance synchronous communications controller (DEUNA) connects VAX and PDP-11 UNIBUS systems to Ethernet local area networks (LANS). This allows large amounts of data to be exchanged at high rates between systems located within limited distances.

The key to this networking feature is Digital's Network Architecture (DNA), a set of hardware and software networking capabilities that supports communications between Digital's systems, and between Digital's systems and other manufacturers' systems. Digital-to-Digital communications are permitted through protocols, or rules, that are defined by the DNA. Internet products provide a means for Digital's systems to communicate with systems built by other manufacturers. These products use common communications protocols and are data transfer facilitators rather than hardware emulators. Digital's packetnet system interface (PSI) products allow Digital's systems to connect to public packet-switching networks.

Digital Networks: An Architecture with a Future and the *Communications Handbook* describe in detail computer networking and the data communications devices available for use with the PDP-11 family members.

▪ Peripherals

The UNIBUS links the PDP-11 processors with one of the industry's most comprehensive set of mass-storage systems. And it provides the configuration flexibility and growth capacity that make the PDP-11 processors ideal solutions for a broad spectrum of applications.

Digital sets the standard for the design and manufacture of storage systems with its Digital Storage Architecture (DSA). DSA is a carefully planned framework of standardized interfaces that permit the addition of new products

and new technologies to host systems without the need to develop additional specialized controllers or software drivers. The PDP-11/84, PDP-11/44, and PDP-11/24 all support the Digital Storage Architecture.

Digital manufactures a full range of peripheral equipment designed to meet specific needs as well as to maintain PDP-11 family compatibility. I/O and storage devices range from low-cost cassette-tape devices through high-capacity Winchester disks, and from intelligent, rugged DECwriters for hard copy, to ergonomically designed video-display terminals. You can also choose from a variety of peripheral products developed and supported by third parties. Either way, there is a complete spectrum of peripherals available to complement the software and provide the complete answer to application needs in all market areas—business, education, industry, laboratory, and engineering environments.

The *Terminals and Printers Handbook* and the *PDP-11 Systems and Options Catalog* describe the optional equipment available for use with the PDP-11 UNIBUS family members.

• Service

Digital's style of computing stresses quality and consistency in every phase of development, manufacturing, service, and support. Comprehensive services are offered to help customers before, during, and after system installation. Our sales representatives work closely with customers, studying each application and determining specific computing needs. Trained software and hardware specialists, well-versed in designing systems using Digital's products, are available to supplement the sales representative's product knowledge.

Comprehensive customer services continue to complete our commitment to meeting users' needs. Digital's customer service organizations include software services, educational services, and field service.

Digital is a complete service vendor and has the products and tools to back its commitment to customer satisfaction. For an indepth description of Digital's PDP-11 UNIBUS systems and PDP-11 software, talk to your Digital sales representative or to one of Digital's commercial or technical OEMs. Or visit one of Digital's Training Centers located throughout the United States. For information on courses, training materials, or training center locations, call the customer support number in Massachusetts: (617) 276-4373.

• Documentation

Digital offers several levels of documentation describing PDP-11 UNIBUS software and hardware. These manuals are updated periodically to include new developments and equipment and can be ordered through Digital's Peripherals and Supplies Group, Continental Boulevard, Merrimack, NH, 03054. The following lists contain the titles and associated Digital order numbers of documents that apply to the PDP-11 UNIBUS family.

Hardware Handbooks and System Manuals

PDP-11 Architecture Handbook	EB-23657-18
Communications Handbook	EB-30066-42
Terminals and Printers Handbook	EB-23909-54
PDP-11/44 System Technical Manual	EK-KD11Z-TM
PDP-11/24 System Technical Manual	EK-11024-TM
PDP-11/84 System Installation and Technical Reference Manual	EK-1184A-TM

Other Technical Manuals and User's Guides

PDP-11/84 Maintenance Guide	EK-PDP84-SV
PDP-11/84 Site Preparation, Unpacking and Installation Guide	EK-PDP84-IN
DCJ11 Microprocessor User's Guide	EK-DCJ-11-UG
KDJ11-B User's Guide	EK-PDP84-UG
MSV11-J MOS Memory User's Guide	EK-MSV1J-UG-001
MSV11-J Configuration Summary	EK-MSV1J-HR-001
PDP-11/44 System User's Guide	EK-11044-UG

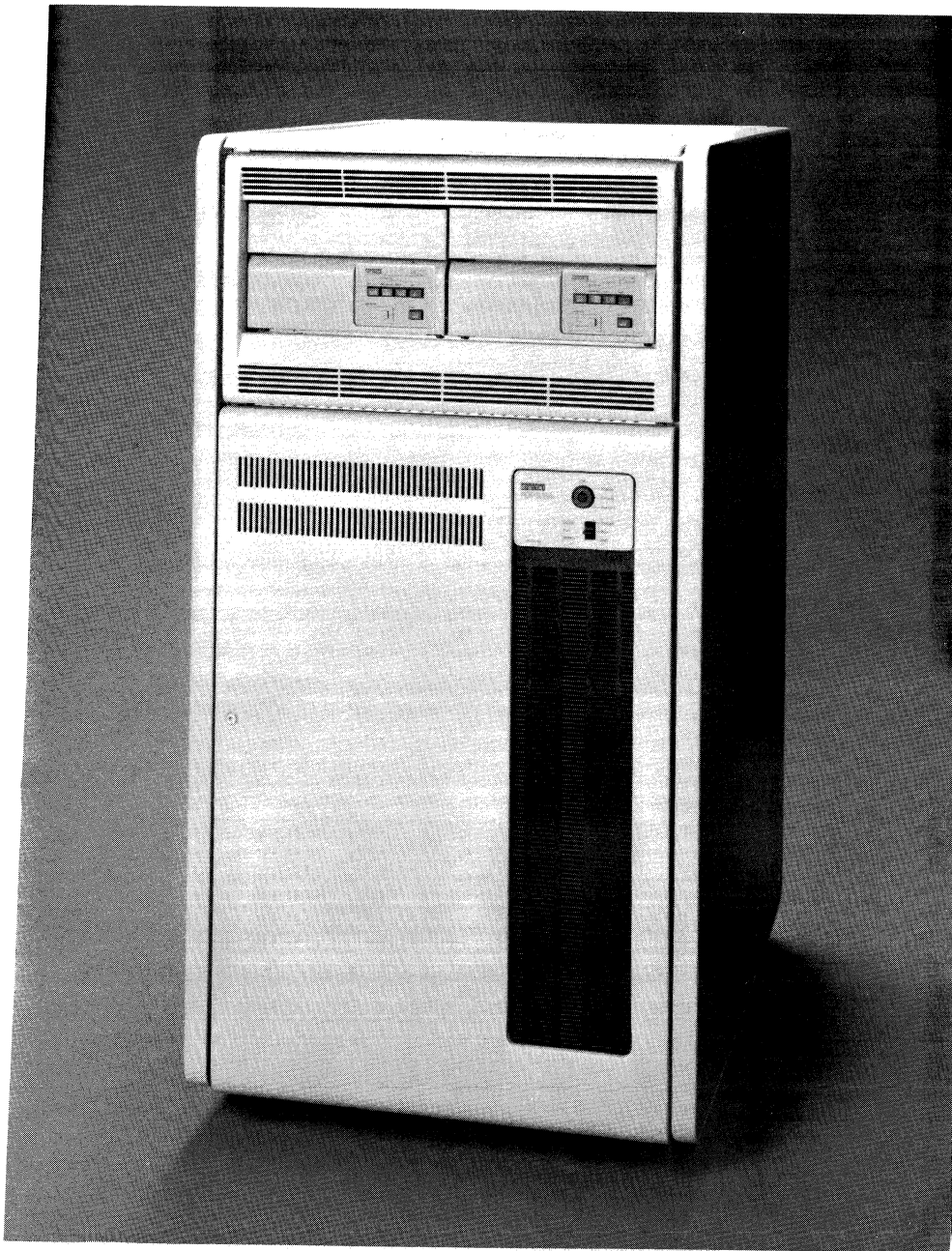
Software Handbooks

PDP-11 Software Handbook	EB-25398-41
RSX-11 Handbook	EB-25742-41
RSTS/E Handbook	EJ-23534-18
ULTRIX Software Guidebook	EJ-26153-20
PDP-11 Software Sourcebook—Fourth Edition (Volumes 1 and 2)	EB-27333-41

Other Important Reference Materials

PDP-11 Systems and Options Catalog	ED-27252-41
Networks and Communications Buyer's Guide	EB-26067-42
Digital's Networks: An Architecture With A Future	EB-26013-42

The hardware user documentation, software tutorial documentation, and reference manuals that accompany the delivery of a PDP-11 computer system offer the most detailed levels of information. There are also several books published commercially that discuss the PDP-11 family. If you have a specific documentation need, discuss the issue with a Digital sales representative who will guide you to the appropriate literature.



The PDP-11/84, the newest member of the PDP-11 UNIBUS family, offers the highest levels of functionality and performance for a machine in its price range. The UNIBUS processor, based on the J-11 chipset and an advanced floating-point processor, surpasses the performance of the PDP-11/70 at about one-third the price. Many new features such as a UNIBUS Adapter (UBA) module with DMA cache and a high-performance private memory interconnect (PMI) that accelerates communications between the CPU and memory are all standard on the PDP-11/84. The 8192-byte high-speed cache memory provides fast access of instructions and data and greatly enhances program execution speeds. The PDP-11/84 handles more users and more memory-resident programs while delivering maximum system throughput. Main memory can be easily expanded to 4 Mbytes. These and other features combine to make the PDP-11/84 the most powerful, yet cost effective UNIBUS PDP-11 system presently offered.

Integral to the PDP-11/84 central processor unit are hardware features and expansion capabilities that are common to the PDP-11/44, PDP-11/24, PDP-11/70 and PDP-11/34A. Table 1-1 illustrates some of the similarities and differences between these earlier PDP-11 UNIBUS systems and the PDP-11/84.

▪ Features

The following are hardware features of the PDP-11/84.

- Powerful CPU features Digital's high-performance J-11 chipset.
- Private memory interconnect (PMI) enhances data transfers between memory, the CPU, and the UNIBUS adapter module.
- High-speed DMA cache provides caching of data for DMA devices.
- 8-Kbyte CPU cache memory speeds program execution.
- Concurrent processing allows simultaneous execution of CPU instructions and UNIBUS I/O data transfers.
- Programmable bus mastership gives the CPU unconditional PMI bus access regardless of pending DMA I/O requests.
- Support of up to 4 Mbytes of error correcting code (ECC) MOS memory.
- Integral floating-point processor operates with 32-bit and 64-bit numbers for faster FORTRAN or BASIC execution.
- Memory management unit provides dual register set, 22-bit addressing, separate instruction (I) and data (D) space, and three operating modes (kernel, supervisor, user).
- Console serial line unit with 8 baud rates (switch-selectable from the rear panel).

- Source control of the line-frequency clock.
- Optional battery backup unit provides power to the memory and air-moving devices in the event of a power failure.
- Standard 32-Kbyte bootstrap and diagnostic ROM resident on the CPU module.
- Mounting space for 4 standard M9312-type UNIBUS bootstrap PROMs to support Digital or customer devices.
- EEROM to set system characteristics and assist user in writing bootstrap programs.

System Architecture

The PDP-11/84 integrates the CPU, the UNIBUS adapter module, and the ECC MOS memory into a general purpose system. Private memory interconnect offers a high-performance communication path between these three modules. A system-level block diagram is shown in Figure 2-1.

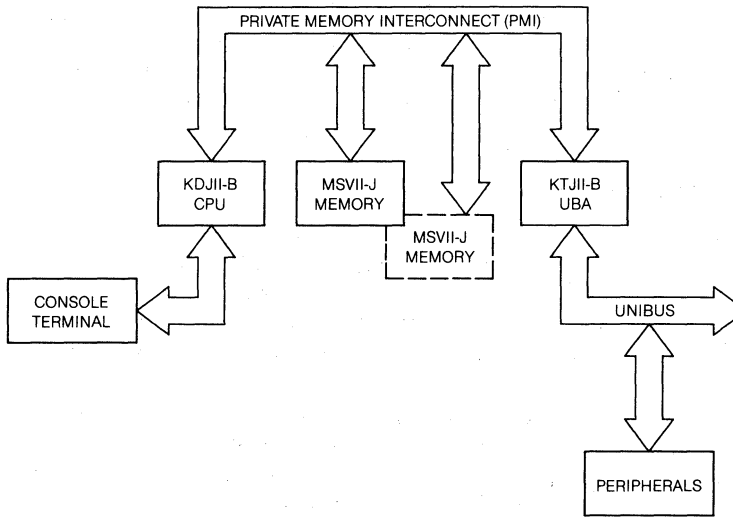


Figure 2-1 ▪ PDP-11/84 System-level Block Diagram

The CPU module comprises the J-11 chipset, floating-point processor, cache memory, a source selectable line-frequency clock, a boot and diagnostic facility, a console serial line unit, and electrically erasable programmable ROMs

(EEPROM). The J-11 microprocessor implements the PDP-11/70 instruction set, including floating-point instructions and memory management. The cache comprises 8-Kbytes of fast, static MOS memory that buffers the processor data from main memory. Dual tag storage enables the PDP-11/84 system to perform CPU cache accesses while concurrently monitoring the cache for DMA hits. Thus, while DMA activity is transferring data on the UNIBUS, the CPU can still access cache memory without being delayed by the DMA. This simultaneous execution makes efficient use of the PMI.

The UNIBUS adapter (UBA) module contains the UNIBUS map for converting between 18-bit UNIBUS addresses and 22-bit PMI memory addresses, the DMA cache, four sockets for M9312-compatible boot ROMs, and the UNIBUS-to-high-speed PMI adapter logic, which interfaces the CPU and PMI memory to the UNIBUS. A block diagram of both the UBA and CPU is shown in Figure 2-2.

The UNIBUS remains the primary I/O control path in the PDP-11/84 system. It is conceptually identical with the UNIBUS of previous PDP-11 systems; the memory in the system still appears to be on the UNIBUS to all UNIBUS devices through the UNIBUS map. There is the additional advantage of the PMI's high-speed communication between the CPU and PMI memory, allowing fast data transfers including double-word reads. The PDP-11/84 system also features programmable bus management, where the CPU can gain bus mastership and compete with DMA devices. The PDP-11/84 can be programmed to occasionally give the CPU unconditional bus priority regardless of how many DMA requests are pending.

▪ Central Processor

The PDP-11/84 CPU module includes the powerful J-11 microprocessor chipset, a floating-point processor, cache memory, a 32-Kbyte bootstrap/diagnostic ROM, a console serial line unit, and a line-frequency clock. Standard features of the J-11 chipset include the floating-point instructions, the PDP-11 instruction set, general purpose registers, processor status word, traps and interrupts, memory system registers, and DMA mechanism.

The J-11 chipset consists of a data chip and a control chip. The data chip performs all arithmetic and logic functions, handles all data and address transfers, and generates most of the signals used for system timing. In addition to the primary execution data path, the data chip contains memory management logic, an I/O state sequencer, and floating-point and cache control registers. The control chip directs the operation of the data chip with microinstructions. The major components of the control chip are the microprogram control store and the microprogram sequencing logic.

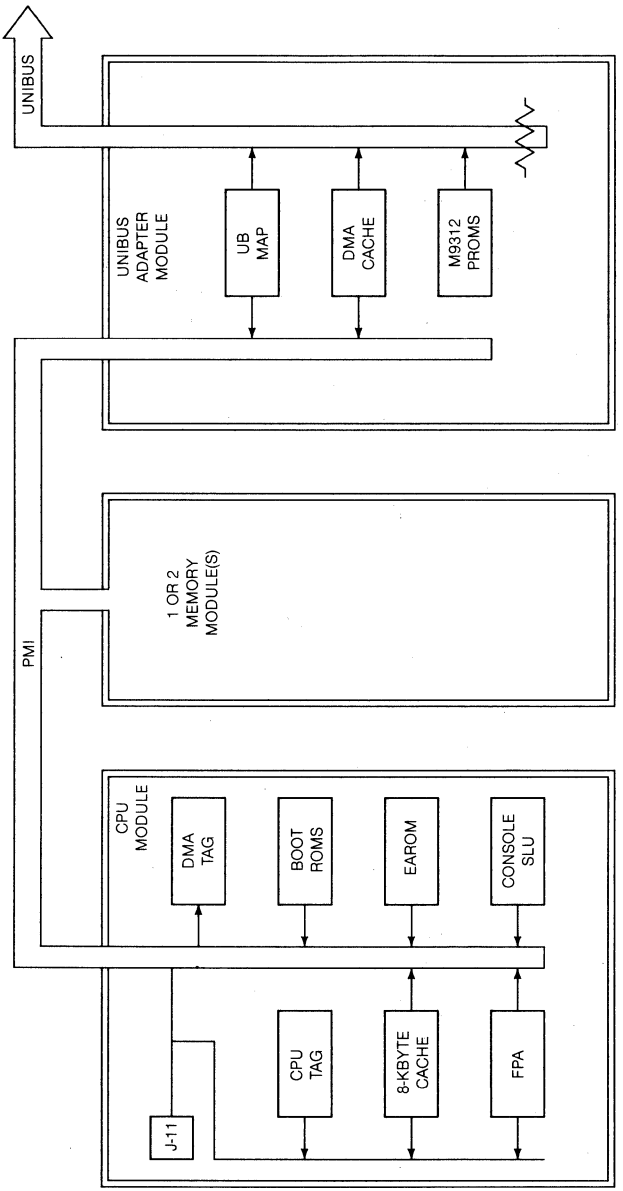


Figure 2-2 ▪ PDP-11/84 CPU and UBA Block Diagram

The machine operates in three modes: kernel, supervisor, and user. When the machine is in kernel mode, a program has complete control of the machine; when the machine is in any other mode, the processor is inhibited from executing certain instructions and can deny direct access to the peripherals on the system. This hardware feature can be used to provide complete executive protection in a multiprogramming environment.

The central processor contains 12 general registers that can be used as accumulators, index registers, or as stack pointers. Stacks are extremely useful for nesting programs, creating reentrant coding, and as temporary storage where a last-in/first-out structure is desirable. An additional register is used as the PDP-11/84's program counter. Three other additional registers are used as processor stack pointers, one for each operational mode.

The CPU performs all of the computer's computation and logic operations in a parallel binary mode through step-by-step execution of individual instructions.

A standard feature of the PDP-11/84 CPU module is the floating-point processor. This coprocessor is a 40-pin chip that significantly improves the computation speed for scientific applications of the CPU module.

General Registers

The general registers can be used for many purposes. Usage varies with requirements. The general registers can be used as accumulators, index registers, autoincrement registers, autodecrement registers, or as stack pointers for temporary storage of data. A chapter on addressing modes in the *PDP-11 Architecture Handbook* describes the uses of general registers in more detail. Arithmetic operations can be performed between one general register and another, from one memory or device register to another, or between a memory or a device register and a general register.

The general registers, as shown in Figure 2-3, include a dual set of six registers (R0 through R5), three stack pointers corresponding to the three processor modes (R6), and a single program counter (R7).

The machine's program counter (PC) contains the address of the next instruction to be executed and thereby controls the order of execution of instructions. The PC is a general register in the sense that it is directly used by all single-operand and double-operand instructions. Much of the power of the PDP-11/84 instruction set is achieved by utilizing the PC in conjunction with various addressing modes. It is a general register normally used only for addressing purposes and not as an accumulator for arithmetic operations.

Register R6 is normally used as the processor stack pointer (SP) indicating the last entry on the current mode's hardware stack, a common temporary storage area with last-in/first-out characteristics. The three stacks are called the kernel stack, the supervisor stack, and the user stack. When the central processor is operating in kernel mode, it uses the kernel stack; in supervisor mode, the

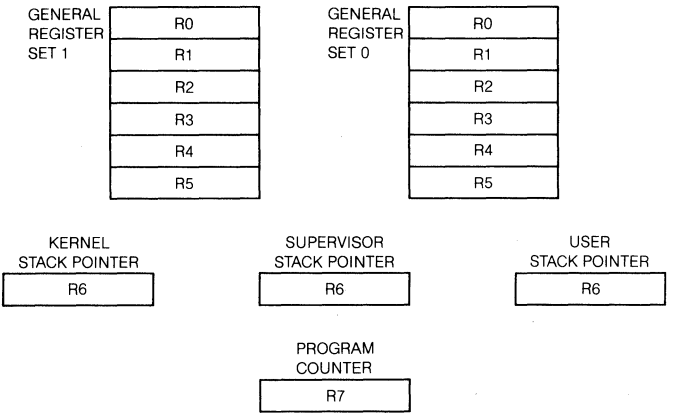


Figure 2-3 ▪ The General Registers

supervisor stack; and in user mode, the user stack. When an interrupt or trap occurs, the PDP-11/84 automatically switches to the mode specified by the service routine and saves its current status on that mode's processor stack. This stack-based architecture facilitates reentrant programming.

The remaining twelve registers are divided into two sets (set 1 and set 0) of unrestricted registers. At any given time, either register set 1 or set 0 is used. The two sets cannot be used simultaneously. The current register set in operation is determined by the processor status word.

The two sets of registers can be used to increase the speed of realtime data handling or facilitate multiprogramming. Each of the six registers in general register set 0 could be used as an accumulator and/or index register for a realtime task or device, or as general registers for a kernel- or supervisor-mode program. The six registers in general register set 1 could be used by the remaining programs or user-mode program. The supervisor or kernel can therefore protect its general registers and stacks from user programs or from other parts of the supervisor or kernel program.

Processor Status Word (PSW)

The processor status word contains information on the current status of the PDP-11. This information includes current and previous operational modes; current register set selection; current processor priority; an indicator for detecting the execution of an instruction to be trapped during program debugging; and the condition codes describing the results of the last instruction. Bit 8 is reserved for future Digital use. Bits 9 and 10 are unused and always read as zeros.

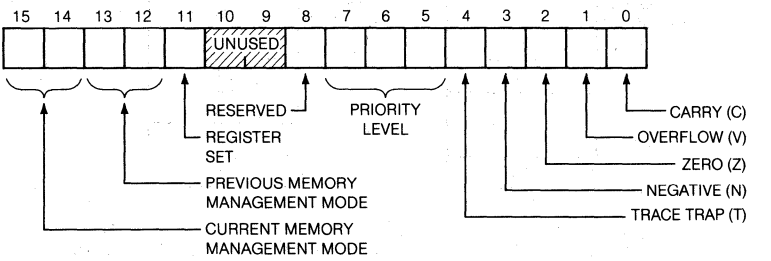


Figure 2-4 ▪ 17 777 776 Processor Status Word

- PROCESSOR MODE FIELDS

Mode information includes the current mode, either kernel, supervisor, or user (bits 15, 14), and the mode the machine was in before the last interrupt or trap (bits 13, 12). The three modes permit a fully protected environment for a multiprogramming system by providing the user with three distinct sets of processor stacks and memory management registers for memory mapping.

In addition, certain PDP-11 instructions are privileged in that their operation is inhibited in supervisor and user modes. For example, in supervisor or user mode, the processor will ignore the RESET and SPL (Set Priority Level) instructions and the HALT instruction will cause a trap through the vector at virtual address 4 in kernel instruction space. In kernel mode, the processor will execute all instructions. A summary of the effects of processor modes on various instruction types is provided in Table 2-1.

Table 2-1 ▪ Instructions Influenced by Processor Modes

	Operation in Kernel Mode	**Operation in Supervisor/User Mode**
HALT	Depends on halt option selected	Traps through a vector at location 4 in kernel data space
WAIT, RESET, SPL	Executes as specified	Executes as a NOP
RTI, RTT, MPTS	Can alter PSW 7–5	Cannot alter PSW 7–5
Stack Reference	Checked for stack overflow	Not checked for stack overflow

- REGISTER SELECTION FIELD

  This one-bit field selects which of the two general purpose register (GPR) sets will be used. When the bit is clear (i.e., 0), GPR set 0 will be used. When the bit is set, GPR set 1 will be used.

- PROCESSOR PRIORITY FIELD

  The central processor operates at any of eight levels of priority, 0–7. When the CPU is operating at level 7, an external device cannot interrupt it with a request for service. The central processor must be operating at a lower priority than the priority of the external device's request in order for the interruption to take effect. The current priority is maintained in processor status word bits 5–7 and is set by the software and used by hardware to determine which interrupts will be processed. The eight processor levels provide an effective interrupt mask, which can be dynamically altered by the kernel-mode program through use of the SPL instruction (used only by the kernel mode). This instruction allows a kernel-mode program to alter the central processor's priority without affecting the rest of the processor status word.

- TRACE TRAP FIELD

  The debugging trace trap is enabled by setting bit 4 of the processor status word and can be set and cleared under program control. When set, a processor trap will occur through location 14 on completion of instruction execution, and a new processor status word and program counter will be loaded. This bit is especially useful for debugging programs because it provides an efficient method of single-stepping the program and is transparent to the general programmer.

  Interrupt and trap instructions both automatically cause the previous processor status word and program counter to be saved and replaced by the new values corresponding to those required by the routine servicing of the interrupt or trap. The user can thus cause the central processor to automatically switch modes (context switching), alter the CPU's priority, or disable the trace trap bit (T) whenever a trap or interrupt occurs.

  NOTE

  Bit 4 of the processor status word can be set only indirectly by executing an RTI (return from interrupt) or RTT (return from interrupt, delaying T-bit trap) instruction with the desired processor status word on the stack. The traced instruction is the instruction after the one that set the T bit.

The following are special cases of the T bit.

- An instruction that clears the T bit—After fetching the traced instruction, an internal flag (trace flag) is set. The trap occurs at the end of this instruction's execution. The status word on the stack, however, will have a clear T bit.
- An instruction that sets the T bit—Because the T bit is already set, setting it again has no effect. The trap will occur.
- An instruction that causes an instruction trap—The instruction trap is performed and the entire routine for the service trap is executed. If the service routine exits with an RTI or in any other way restores the stacked status word, the T bit is set again, the instruction following the traced instruction is executed and, unless it is one of the special cases noted previously, a trace trap occurs.
- An instruction that causes a stack overflow—The instruction completes execution as usual. The stack overflow does not cause a trap. The trace trap vector is loaded into the program counter and the processor status word, and the old program counter and processor status word are pushed onto the stack. Stack overflow occurs again, and this time the trap is made.
- An interrupt between setting the T bit and fetching the traced instruction—The entire interrupt service routine is executed and then the T bit is set again by the exiting RTI. The traced instruction is executed (if there are no other interrupts) and, unless it is a special case noted above, causes a trace trap.

- CONDITION CODES FIELD
  The condition codes contain information on the result of the last CPU operation. They include a negative bit (N), set if the result of the previous operation was negative; a zero bit (Z), set if the result of the previous operation was zero; an overflow bit (V), set if the result of the previous operation resulted in an arithmetic overflow; and a carry bit (C), set if the result of the previous operation caused a carry out of the most-significant bit.

Program Interrupt Request Register (PIRQ)

The program interrupt request (PIRQ) register (see Figure 2-5) provides seven levels of software interrupt capability. An interrupt request is queued by setting one of bits 15–9, which corresponds to interrupt priority levels 7 through 1, respectively. Bits 7–5 and 3–1 are set by the J-11 microprocessor to the encoded value of the highest pending request. When the program interrupt request is granted, the processor traps through the vector at virtual location 240. It is the responsibility of the interrupt service routine to clear the appropriate bit in the PIRQ before the interrupt is dismissed.

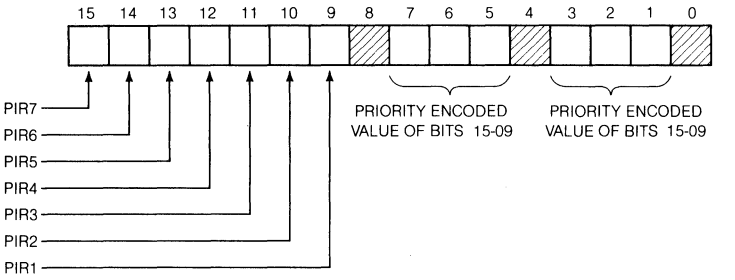


Figure 2-5 ▪ 17 777 772 Program Interrupt Request Register

Because the PIRQ interrupt can be caused by any priority between 7 and 1, the PIRQ service vector usually causes the processor to service the PIRQ interrupt at priority level 7 (this locks out further PIRQ interrupts). Bits 7–5 provide a convenient way of lowering the processor's priority to the level that actually caused the PIRQ interrupt. For example, if a level-5 PIRQ interrupt occurs, the CPU hardware would set bits 7–5 = 240. Putting this value in the PSW would place the CPU at priority level 5, blocking further level 5 interrupts.

Bits 3–1 once again provide the actual priority of the PIRQ interrupt. This time the bit field is positioned to allow its use as a word-offset (e.g., bits 3–1 could be used with an indexed-JMP instruction to dispatch each PIRQ interrupt level to a unique address).

Bits 15–9 are read or write. Bits 7–5 and 3–1 are read-only. The remaining bits are always read as zeros. PIRQ is cleared by a console start, by a RESET instruction, or at powerup time.

Pipeline Processing

The J-11 chipset gets much of its performance from its prefetch and predecode mechanisms. The primary benefit of prefetch and predecode is that memory references are overlapped with internal operations, and the need for explicit instruction fetch and decode cycles is minimized. The prefetch and predecode operations are performed automatically by the microprocessor and cannot be altered by the user.

A primary function of the prefetch mechanism is to fill four registers with information and replenish the registers as required. These four registers, the virtual program counter (VPC), the physical program counter (PPC), the prefetch buffer (PB), and the instruction register (IR) are collectively referred to as the prefetch pipeline. The contents of registers in the beginning of the pipeline are used to determine the contents of registers farther down the

pipeline. This four-stage pipeline processing enables the processor to overlap execution when decoding, addressing, and fetching instructions. When the pipeline is filled, the prefetch mechanism is said to be in steady state.

Once the prefetch mechanism is in steady state, a stream of macroinstructions that operate only on registers may be executed at the rate of one per microcycle. While one instruction is being executed, the next one is being decoded, and the following one is being prefetched into the PB. This maintains the steady state, allowing the next macroinstruction to be executed in the next microcycle. The J-11 bus is kept busy 100 percent of the time.

The instructions that operate on immediate data and a register also make maximum use of the prefetch mechanism. At steady state, a stream of the macroinstructions executes in two microcycles. At the same time the operation is being performed, the data in the PB is being moved to a scratch register. In both cycles, the steady state of the prefetch mechanism is maintained by prefetching the next instruction stream word.

The prefetch pipeline is refilled after a powerup sequence or if a prefetch fault occurs. Prefetch faults occur when the PSW, cache control register, PC or any other memory management registers are written. A prefetch fault invalidates only the PB. This means that the pipeline remains synchronized and can be refilled in two microcycles.

CPU Error Register

The CPU error register (Figure 2-6) assists the operating system by identifying the source of the abort that caused a trap through the vector at location 4.

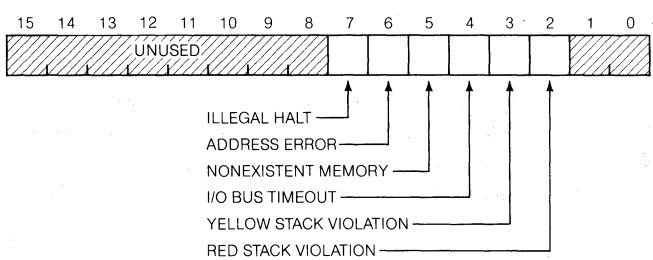


Figure 2-6 ▪ 17 777 766 CPU Error Register

Bits: 15–8
Name: Unused
Function: These bits are unused and are always read as zeros

Bit: 7
Name: Illegal HALT (Read Only)
Function:Set when execution of a HALT instuction is attempted in user or supervisor mode, or in kernel mode when the halt-trap option is enabled.

Bit: 6
Name: Address Error (Read Only)
Function:Set when a word access is made to an odd byte address, or when an instruction fetch from an internal register is attempted.

Bit: 5
Name: Nonexistent Memory (Read Only)
Function:Set when reference is made to a nonexistent memory address.

Bit: 4
Name: I/O Bus Timeout (Read Only)
Function:Set when reference is made to a nonexistent I/O page address.

Bit: 3
Name: Yellow Stack Trap (Read Only)
Function: Set when a yellow zone stack overflow trap occurs.

Bit: 2
Name: Red Stack Trap (Read Only)
Function: Set when a red zone stack overflow trap occurs.

Bits: 1–0
Name: Unused
Function: These bits are unused and are always read as zeros.

The CPU error register is cleared by any write reference to it, by a powerup, or by a console start. The RESET instruction has no effect on this register.

Stack Limit Protection

The J-11 microprocessor provides hardware protection for the kernel stack. The supervisor and user stacks are not protected by hardware, but may be checked by memory management and appropriate software.

Stack protection in kernel mode is provided by defining yellow and red stack traps. Kernel stack references are checked against a fixed limit of 400 (octal). If the virtual address of the stack reference is less than 400 (octal), a yellow stack trap occurs at the end of the current instruction. The PDP-11/84 treatment of a yellow stack trap is identical to that of the PDP-11/44 treatment of yellow stack trap.

The J-11 chipset also checks for kernel stack aborts during interrupt, trap, or abort sequences. If an abort is caused by a kernel stack push during an interrupt, a trap, or an abort sequence, the J-11 initiates a red zone stack trap by setting CPU error register bit 2, loading virtual address 4 into the kernel stack pointer, R6, and trapping through location 4 in kernel data space. The J-11 microprocessor's definition of a red stack trap is unique.

Kernel Protection

In order to protect the kernel operating system against interference, the J-11 microprocessor incorporates a number of protection mechanisms.

- Limiting execution of certain instructions.
- Limiting user-program access to data.
- Limiting user-program modification of data.
- Limiting user-program access to I/O.

- INSTRUCTION LIMITATION

As mentioned previously, only a kernel-mode program may execute certain instructions. These instructions are

- SPL—Set processor priority level
- RESET—Initialize the I/O system.
- HALT—Halt the entire PDP-11/84.

If a user- or supervisor-mode program attempts to execute SPL or RESET, nothing happens. If a user- or supervisor-mode program attempts to execute a HALT instruction, a trap is taken. This allows the operating system to simulate halts while in a multiuser environment.

In addition, certain other instructions execute differently while in kernel mode.

- MFPD—Move from previous data space.

- ADDRESSING LIMITATIONS

The operating system can limit the amount of physical memory visible to any of the three modes of operation. Typically, only the kernel-mode program has access to the I/O page. Because the registers which control access to memory are themselves located in the I/O page, this means that only the kernel-mode program can change the access rights of the other modes.

Memory is allocated by pages to each of the operating modes. Each operating mode allocates 8 pages for code and 8 pages for data. Each page may be from zero to 8,192 bytes long, in increments of 64 bytes. Each page may be marked as read/write or read-only.

Taken together, these capabilities provide a robust protection scheme which ensures the integrity of a multiprogramming system. The hardware features are used extensively by the RSX, RSTS/E, IAS and ULTRIX multiprogramming operating systems.

Figure 2-7 illustrates how the user's "view" of the entire system can be limited (much different from the kernel's "view"). The user has no ability to access anything outside of the user's area of memory. This gives complete protection.

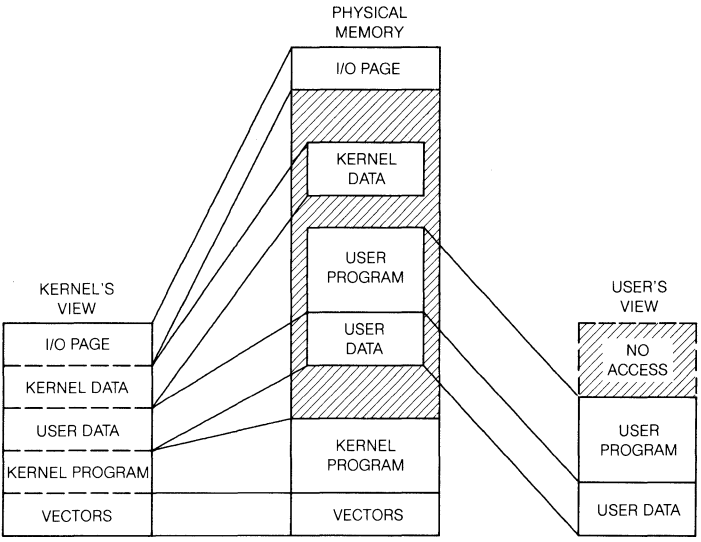


Figure 2-7 ▪ Kernel's and User's View of Physical Memory

Trap and Interrupt Service Priorities

Interrupts and traps are requests that cause the PDP-11/84 to suspend the execution of the current program temporarily and provide service for the device or condition that caused the interrupt or trap. Interrupts differ from traps in that interrupts are initiated asynchronously by some external event while traps are caused by synchronous or asynchronous conditions internal to the PDP-11/84.

When an interrupt or trap occurs, the current PSW and PC are preserved in order to allow a return to the interrupted program. The new contents of the PC and the PSW are fetched from two consecutive memory words called a vector. The first word of the vector contains the interrupt or trap service routine starting address (the new PC), and the second word contains the new PSW. Some vectors are predefined by the PDP-11/84 while others are user defined.

The priority order for traps and interrupts is as follows.

Red stack trap
Address error
Memory-management violation
Timeout/nonexistent memory
Parity error
Trace (T-bit) trap
Yellow stack trap
Powerfail
Floating-point trap
PIRQ 7
Interrupt level 7
Line-time clock
PIRQ 6
Interrupt level 6
PIRQ 5
Interrupt level 5
PIRQ 4
Interrupt level 4
PIRQ 3
PIRQ 2
PIRQ 1
Halt line*

Hardware Detected Errors

The PDP-11/84 detects certain hardware error conditions during program execution. These conditions, and the resultant actions, are described in the following paragraphs.

- BUS TIMEOUT ERROR

A bus timeout error can occur during PDP-11/84 memory or UNIBUS cycles, when a slave device fails to respond. This generally occurs when the addressed memory and I/O device do not exist in the particular system. The processor recognizes the error condition and immediately traps through virtual address 4 in the kernel data space. The UNIBUS Adapter module asserts the PMI timeout signal causing the CPU to abort.

*The halt line is given highest priority when the processor hangs up.

- ADDRESSING ERROR

  An addressing error occurs when an odd byte address is used with a word reference (odd addressing error) or an instruction stream fetch attempts to access an internal processor register. The internal processor registers include PARs, PDRs, CPU error, processor status word, program interrupt request, MMR0-MMR3, hit/miss, and cache control. When an addressing error happens, it sets bit 6 of the CPU error register and traps through virtual address 4 of the kernel data space.

- RED STACK ABORTS

  A red stack abort occurs if, during the servicing of an abort, interrupt or trap routine, an abort occurs while pushing the processor status word or program counter onto the kernel stack. This type of abort sets bit 2 of the CPU error register, loads the kernel stack pointer (R6) with virtual address 4, and then traps through location 4 of the kernel data space. The last PC and PSW are saved in locations 0 and 2 of the kernel data space.

• Private Memory Interconnect (PMI)

The UNIBUS is a general purpose bus designed to handle numerous interface modules and a variety of peripheral devices. The UNIBUS protocols were designed for optimum data transfer on a 50-foot long bus. This length and the delay characteristic of the associated bus drivers and receivers result in a relatively low data transfer rate. To increase data transfer rate on the PDP-11/84 system, Digital's engineers had to extend and upgrade the existing bus structure so that it could function concurrently as two separate buses. The result was the development of the private memory interconnect (PMI), which shares some of the existing bus hardware, but at the same time has its own bus architecture and data transfer protocols.

The PMI's principal contribution is to double the speed of DMA and CPU/memory transfers. The basis for this process is serial double-word transfers, often called double-pumping. Double-word transfers involve moving two words in sequence with only one address. Performance improvement was achieved, first, with a private memory interconnect having faster drivers and receivers and accommodating the reduced transfer delay demands on a shorter bus. The interconnect's transfer protocols then take advantage of the high-speed capability.

The PMI consists of a multiplexed data/address commmunication path between the CPU, PMI memory, and the UNIBUS adapter module (UBA). Direct access to the PMI is reserved exclusively for these three modules (see Figure 2-2) All communication between these modules and the UNIBUS devices is controlled and directed through the UBA module. The CPU and UBA

modules, but not the memory module, can become PMI master. All three can respond as PMI slaves. The CPU is always the PMI default master and the UBA is the UNIBUS's default master. The PMI arbitration logic is located on the CPU and arbitrates both PMI and UNIBUS interrupt/data transfers.

Once a bus structure is established, its effective speed in memory transfers depends on the bus protocols and control signals that make it work. In the PMI, the control signals manage PMI operations to reduce total transfer time. Twenty control signals are used in the PMI protocol.

The PMI supports standard UNIBUS interrupt transfers through the UBA. During a vectored interrupt transfer, the UNIBUS device initiates the interrupt request line that is then received directly by the processor logic. After receiving the interrupt request, the processor arbitration logic arbitrates priorities and grants PMI mastership to the UBA interrupting device. After acquiring bus mastership, the UNIBUS device can control data or vector interrupt transfers via the UBA.

▪ Memory System

Memory Management

Memory management hardware is standard on the PDP-11/84 system, providing a dual register set, 22-bit addressing, separate instruction and data space, and three operating modes. The J-11 chipset contains the memory management unit (MMU) that provides the user with the hardware necessary to effect complete memory management and protection. The MMU is designed to provide access to all of physical memory and is an important part of multiuser, multiprogramming systems on which memory protection and relocation facilities are necessary.

The basic characteristics of the PDP-11/84 memory management unit are

- 16 kernel-mode memory pages.
- 16 supervisor-mode memory pages.
- 16 user-mode memory pages.
- 8 pages in each mode for instructions.
- 8 pages in each mode for data.
- Page lengths from 64 to 8,192 bytes, in increments of 64 bytes.
- Full protection and relocation for each page.
- Transparent operation.
- Memory access to 4 Mbytes.

The PDP-11/84 implements PDP-11/44 compatible memory management. The visible memory management state consists of 48 page address registers (PARs), 48 page descriptor registers (PDRs), and 4 memory management registers (MMRO, MMR1, MMR2, MMR3). Details of these registers may be found in both the *KDJ11-B User's Guide* and the *PDP-11/84 Technical Manual.*

The PDP-11/84 processor can perform 16-bit, 18-bit, or 22-bit address mapping. This mapping operation provides compatibility with the other PDP-11 computers. This means that software written and developed for any PDP-11 computer can run on the PDP-11/84 without modification.

Error Correction Code (ECC)

Error correction code (ECC) is a technique used to check the contents of memory to detect errors and correct them before sending the data to the processor. The process of checking is accomplished by combining the bits in a number of unique ways so that parity, or syndrome, bits are generated for each unique combination and stored along with the data bits in the same word. The memory word length is extended to store these unique bits. When memory is read, the data word is checked against the syndrome bits stored with the word. If they match, the word is sent on to the processor. If they do not match, an error exists and the mismatch of the syndrome bits determines which data bit is in error. The bit in error is then corrected and this corrected data is then sent on to the processor. The ECC that is employed in MOS memory will detect and correct single-bit errors in a word, and detect double-bit errors in a word. Where a double-bit error is detected the data cannot be corrected so the processor is notified (as happens with a parity error in parity memory).

ECC provides maximum system benefits when used in a storage system that fails in a random single-bit mode rather than in blocks or large segments. Single-bit errors (or failures) are the predominant failure mode for modern MOS memories.

ECC memory provides fault tolerance with the result that multiple hard or soft single-bit failures can be present in a memory system without causing measurable degradation in either performance or reliability.

Battery Backup Unit

MOS memory is volatile. It depends on electricity to store information. Because a power loss or shutdown causes data loss, battery backup units (BBU) are designed to temporarily preserve data in memory. These units are available as options on the PDP-11/84 system.

Generally, the incidence of ac line power loss varies inversely with the severity of loss. That is, there are an extremely small number of long-term failures of ac power and a relatively larger number of short-term failures or drops in voltage.

Battery backup units are not intended to preserve data overnight or over weekends, but rather to prevent data loss during infrequent, short-term failures of ac power.

UNIBUS Adapter (UBA)

The UNIBUS Adapter (UBA) is essentially a special-purpose DMA device that contains the UNIBUS to PMI adapter logic, UNIBUS mapping, and four boot ROM sockets. The multifunction UNIBUS Adapter links the PMI and UNIBUS, to which is connected one or more UNIBUS-compatible peripheral devices. The UBA communicates with the CPU and memory via the PMI protocol and communicates with peripheral devices on the UNIBUS via the UNIBUS protocol. Among its many functions, the UBA is responsible for translating the 18-bit UNIBUS DMA addresses to 22-bit PMI addresses and vice-versa exchanging information between UNIBUS and PMI protocols.

- DMA CACHE

Performance of UNIBUS DMA is improved by adding a 32-word cache to the UBA module (see Figure 2-8). When a DMA read from memory occurs and the address is an even 8-word boundary, the first word of data is transferred to the I/O device and the next 7 words are cached. When the data from the next 7 addresses is required, this data is taken directly from the cache. The DMA cache is a four-set associative cache. DMA cache contains 32 16-bit data registers, arranged in four sets of eight data registers each. Associated with each set is a valid bit and an 18-bit tag register. The data registers are located in RAM memory. The tag registers and valid bits are located in the UBA gate array. The operation of DMA cache is transparent to the software.

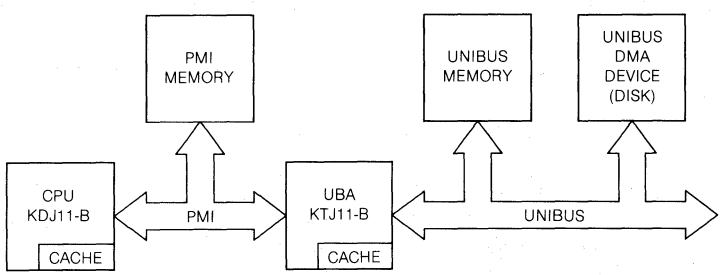


Figure 2-8 ▪ PDP-11/84 Cache Diagram

The DMA cache allows the UBA to "group" UNIBUS word read-operations into PMI octal-words. It also provides a speed-matching buffer to level out variations in PMI or UNIBUS traffic. Since the cache is organized as 4 groups of 8 words each, the UBA can very efficiently manage up to 4 concurrent UNIBUS DMA devices without thrashing.

- UNIBUS MAPPING

  The UNIBUS map is the interface between the UNIBUS and PMI memory. It responds as a slave to UNIBUS masters and is used to convert 18-bit UNIBUS addresses to 22-bit memory addresses. The 22-bit memory address is accompanied by an additional signal line, BBS7 L. The assertion of BBS7 L disables the PMI address decoding and selects the I/O page.

  The UNIBUS address space is 256 Kbytes, of which the top 8,192 (8-Kbyte) addresses always refer to the I/O page. The lower 248 Kbytes of UNIBUS address space can be used by the UNIBUS map to refer to physical memory. The UNIBUS map can be programmed, via memory management register 3 (MMR3) to run with relocation enabled or disabled. UNIBUS mapping and the memory management registers are explained in detail in the *PDP-11 Architecture Handbook*.

  The ability to install UNIBUS memory instead of, or in addition to, PMI memory has been preserved with the PDP-11/84. UNIBUS address space is assigned to UNIBUS memory in 8-Kbyte segments, starting with the segment below the I/O page and proceeding downward. The UNIBUS address segments assigned to UNIBUS memory cannot be used to access PMI memory via the I/O map. Whenever the CPU accesses UNIBUS memory, the UBA will disable the PMI memory response. The CPU module does not cache UNIBUS memory.

  The Memory Configuration Register (KMCR), one of three registers on the UBA, reflects the placement of UNIBUS memory by allowing the CPU boot and diagnostic programs to configure the UBA for the distribution of UNIBUS and main memory within the system. Additional KMCR bits allow the DMA cache to be enabled and disabled, provide diagnostic status of the read buffer and supply information on the reboot status of the system. For a more detailed description of the optional UNIBUS memory and the registers on the UBA refer to the *PDP-11/84 Technical Manual*.

- BOOT ROM FACILITY (M9312 COMPATIBLE)

  The UBA boot ROM facility allows the user to install M9312-compatible boot programs written for UNIBUS devices that are not directly supported by the CPU boot programs. ROM programs that run on the M9312 also work on the UBA. The M9312-compatible boot programs are implemented with from one to four 512-by-4-bit ROMs. Each ROM contains 64 16-bit words of accessible code that is located in the first half of the ROM. The last 256 4-bit ROM locations are not used.

The CPU module can be configured to boot the system from

- one of its self-contained boot programs.
- EEROM, user written in EEROM.
- the UBA boot ROM facility.
- a M9312 type ROM option that resides on the UNIBUS.

For details of the location of each ROM socket along with its address range, see the *PDP-11/84 Technical Manual.*

Cache Memory

The PDP-11/84 has a large 8-Kbyte cache located on the CPU module that is used to decrease the system access time of instructions and data (see Figure 2-8). CPU cache operations occur only for memory on the PMI bus. Memory on the UNIBUS, if present, is not cached.

The 8-Kbyte write-through direct map CPU cache has dual tag stores that allow concurrent operations of the CPU and DMA. The cache is transparent to all programs and acts as a high-speed buffer between the processor and PMI memory. The data stored in cache represents the most active portion of the PMI memory being used. The processor accesses main memory only when data is not available in the cache.

Cache memory is a high-speed memory that buffers data between the CPU and main memory. When a memory access occurs, the system looks first for data in cache memory. If found (a hit), the data is read from cache and execution proceeds at the fastest rate. If not found (a miss), the data must be read from memory and written to cache.

In a write-through cache system, such as the PDP-11/84, a CPU request to write data into memory causes data to be written to both the cache and to main memory. This ensures that both stores are always updated immediately.

Typical hit/miss ratios in a write-through cache system are summarized in Table 2-2. In a typical program, WRITEs occur only 10–15 percent of the time and READs occur 85–90 percent of the time. Thus, READ misses cause the cache to be updated.

Table 2-2 • Typical Hit/Miss Operations

	CACHE	MAIN MEMORY
READ		
hit	no change	no change
miss	updated	no change
WRITE		
hit	updated	updated
miss	no change	updated

The top 8 Kbytes of physical memory (the I/O page) is not cached. This is because the I/O page contains device status registers that, when read, must always convey the latest information.

When a DMA device writes to a cached location, the cache entry must be invalidated. The cache system monitors DMA transactions to determine if this action is needed. Because the PDP-11/84 contains two identical tag stores, this DMA monitoring can overlap CPU cache accesses. Only if a DMA write-hit occurs must the CPU be stopped to invalidate the cached location.

The following matrix (Table 2-3) shows the cache response for both DMA and CPU data transfers from and to the PMI memory space. There are two cache tag memories refered to in the matrix. The DMA tag matrix heading refers to DMA activity. The CPU tag matrix heading refers to PMI activity involving the CPU.

Cache parity errors affect the cache response matrix in the following ways.

- During DMA write cycles, a DMA tag parity error forces a cache hit response, and the cache location is invalidated.
- During CPU read cycles (nonbypass), a CPU tag or data parity error forces a cache miss response.
- During CPU write-byte cycles (nonbypass; nonforce miss), a CPU tag parity error forces a cache hit response, but the data is loaded with bad parity.
- During CPU read bypass or write bypass cycles, a CPU tag or data parity error forces a cache hit response. The cache location is invalidated.
- For all force miss cycles, and for the CPU write word (nonbypass) cycle, cache parity is ignored.

Cache Registers

The PDP-11/84 contains hardware that allows the user to control the cache memory. This hardware consists of the cache control register, the memory system error register, and the hit/miss register. These registers allow for a broad spectrum of cache implementations and considerable flexibility in designing a cache memory scheme to fit a specific application.

Table 2-3 ▪ PDP-11/84 Cache Response Matrix

▪ DMA TAG

	Hit	Miss
Read	Read memory	Read memory
Write word	Invalidate cache Write memory	Write memory
Write byte	Invalidate cache Write memory	Write memory
Read bypass	—	—
Write bypass	—	—
Read force miss	Read memory	Read memory
Write force miss	Write memory Invalidate cache	Write memory

▪ CPU TAG

	Hit	Miss
Read	Read cached data	Read memory and allocate cache
Write word	Write both cache and memory	Write memory No cache change
Write byte	Write both cache and memory	Write memory No cache change
Read bypass	Invalidate cache and read memory	Read memory No cache change
Write bypass	Invalidate cache and write memory	Write memory No cache change
Read force miss	Read memory No cache change	Read memory No cache change
Write force miss	Write memory No cache change	Write memory No cache change

- CACHE CONTROL REGISTER (CCR)

Bits: 15–11
Name: Unused
Function: These bits always read as zeros.

Bit: 10
Name: Wrong Tag Parity
Function: When this read/write bit is set, the CPU and DMA tag parity bits are both written as wrong parity during all operations that update these bits. A cache tag parity error will thus occur on the next access to that location.

Bit: 9
Name: Unconditional Cache Bypass
Function: When this read/write bit is set, all references to memory by the CPU will bypass the cache and go directly to main memory. Read or write hits will result in the invalidation of the corresponding cache location; misses will not affect the cache contents.

Bit: 8
Name: Flush Cache
Function: Writing a one into this write-only bit clears all CPU tag and DMA tag valid bits invalidating the entire contents of the cache. Writing a zero into this bit has no effect. Flush Cache always reads as zero. The CPU requires approximately 1.2 milliseconds to flush the cache.

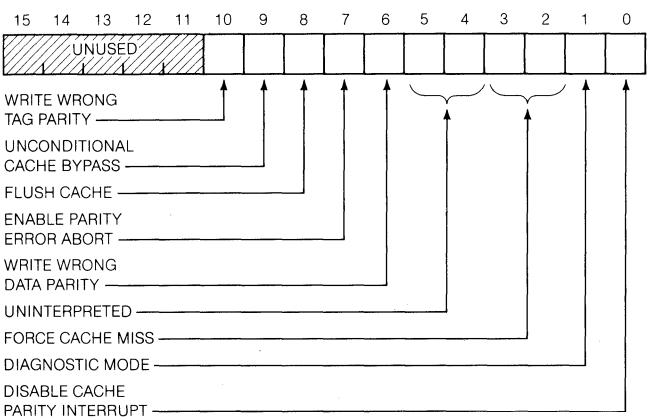


Figure 2-9 ▪ 17 777 746 Cache Control Register

Bit: 7
Name: Parity Error Abort
Function: This read/write bit is set for diagnostic purposes only. When it is set, a cache parity error (during a CPU cache read) will cause the CPU to abort the current instruction and trap to parity error vector 114. When this bit is clear, a cache parity error (during a CPU read) results in a force miss and data fetch from main memory. The CPU will trap to 114 only if CCR bit 0 is clear. DMA cycle cache parity errors will cause a trap to 114 if CCR 7 is set or CCR 0 is clear. CCR 7 has no effect on main memory parity errors that always cause the CPU to abort the current instruction and trap to 114.

Bit: 6
Name: Write Wrong Data Parity
Function: When this read/write bit is set, both the high and low data parity bits are written with wrong parity during all operations that update these bits. This will cause a cache data parity error to occur on the next access to that location.

Bits: 3–2
Name: Force Miss
Function: When either of these read/write bits is set, CPU reads will be reported as cache misses.

Bit: 1
Name: Diagnostic Mode
Function: When this read/write bit is set, a 10-microsecond nonexistent memory timeout during a word write will not cause a nonexistent memory trap and will not set CPU error register bit 5. All nonbypass and nonforced miss word writes will allocate the cache regardless of the nonexistent memory timeout.

Bit: 0
Name: Disable Cache Parity Interrupt
Function: This read/write bit controls cache parity interrupts when CCR 7 is clear (normal operation). If CCR 7 is clear, a cache parity error (during a CPU cache read) results in a force miss and data fetch from main memory. The CPU will trap to 114 only if CCR bit 0 is clear. DMA cycle cache parity errors will cause a trap to 114 if CCR 7 is set or if CCR 0 is clear.

- MEMORY SYSTEM ERROR REGISTER (MSER)

Bit: 15
Name: CPU Abort
Function: This read-only bit is set if a cache or main memory parity error results in an instruction abort, such as during the demand read cycle. Cache parity errors cause an abort only if CCR 7 is set. Main memory parity errors always cause an abort.

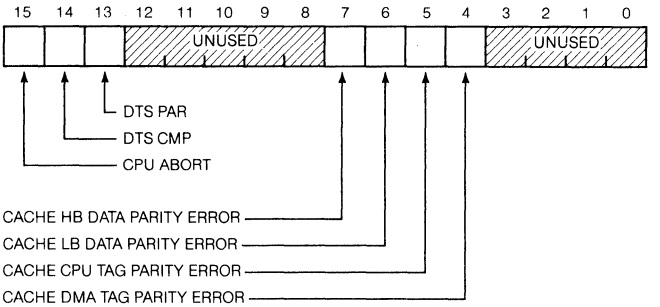


Figure 2-10 ▪ Memory System Error Register

Bit: 14
Name: DMA Tag Store Comparator (DTS CMP)
Function: In standalone mode (BCSR bit 8 set), this read-only bit indicates the output of the cache DMA tag store comparator for the previous non-I/O page reference with cache miss. When BCSR 8 is clear, DTS CMP reads as zero.

Bit: 13
Name: DMA Tag Store Parity (DTS PAR)
Function: In standalone mode (Boot and Diagnostic Controller Status Register (BCSR) bit 8 set) this read-only bit indicates the output of the DMA tag store parity check logic for the previous non-I/O page reference with cache miss. When BCSR 8 is clear, DTS PAR reads as zero.

Bits: 12–8
Name: Unused. These bits always read as zeros.

Bit: 7
Name: Cache HB Data Parity Error
Function: This read-only bit is set if a parity error is detected in the high-data byte during a CPU cache read. If CCR 7 is clear, MSER 7 is also set by a low-data byte parity error and by the set condition of MSER 5 or 4.

Bit: 6
Name: Cache LB Data Parity Error
Function: This read-only bit is set if a parity error is detected in the low-data byte during a CPU cache read. If CCR 7 is clear, MSER 6 is also set by a high-data byte parity error and by the set condition of MSER 5 or 4.

Bit: 5
Name: Cache CPU Tag Parity Error
Function; This read-only bit is set if a parity error is detected in the CPU tag field during a CPU cache read. If CCR 7 is clear, MSER 7 is also set by a high- or low-data byte parity error.

Bit: 4
Name: Cache DMA Tag Parity Error
Function; This read-only bit is set if a parity error is detected in the DMA tag field during a DMA write operation.

NOTE

Cache parity errors are ignored (that is, they do not affect MSER 7–4) if either CCR 3 or 2 (force miss) is set or if the CPU tag valid bit is clear.

Bits: 3–0
Name: Unused. These bits always read as zeros.

- HIT/MISS REGISTER

This register indicates whether the six most recent CPU memory references resulted in cache hits or cache misses. Bits enter from the right (at bit 0) and are shifted left. A one indicates a cache hit, a zero indicates a cache miss.

The hit/miss register is read-only. Bits 15–6 are not used and always read as zero. The register's value at powerup is undefined and is not affected by console start or a RESET instruction. The hit/miss register will always read zero when the CPU is in console ODT mode.

Figure 2-11 ▪ Hit/Miss Register

▪ Backplane

The PDP-11/84 backplane (Figure 2-12) consists of 13 slots. Slot MDM is reserved for the monitor and distribution module (MDM). Slot 1 is dedicated for the KDJ11-B CPU module. Slots 2 and 3 are reserved for the MSV11-J memory modules. If only one memory module is installed, either slot 2 or 3 can be used. The final dedicated slot, slot 4, is for the UNIBUS adapter module. Slots 1 through 4 are connected via the PMI bus on rows A, B, C, and D. Slots 5 through 12 are wired to support any UNIBUS-compatible, small peripheral controller (SPC). Quad SPCs are installed in rows C, D, E, and F. In addition to SPC support, slots 9, 10, and 11 support modified UNIBUS devices. A terminator module or UNIBUS out cable must be installed in slot 12, rows A and B.

The monitor and distribution module has unique features that are essential to the function of the PDP-11/84. It is a quad-height board and includes power supply voltage indicators and test points, fan/blower rotation monitor with

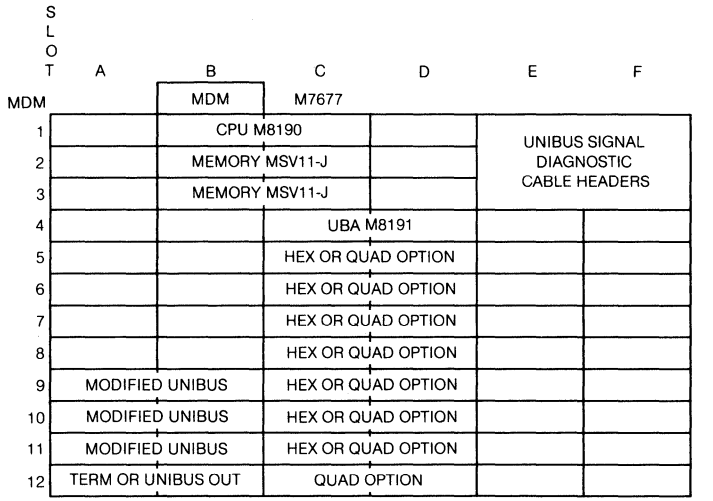


Figure 2-12 ▪ PDP-11/84 Backplane Configuration

audible alarm that shuts down the system after 60 seconds of air-moving failure, and nonprocessor jumper switches for slots 5 through 12. It has module fingers for row B only and occupies only backplane slot MDM.

▪ Console Functions

Console Serial Line Unit

The console serial line unit provides the J-11 microprocessor with a serial interface for the console terminal. There are four console serial line unit registers—the receiver status register, the receiver data buffer, the transmitter status register, and the transmitter data buffer. These registers cannot be disabled.

▪ RECEIVER STATUS REGISTER (RCSR)

Bits: 15–12
Name: Unused. They always read as zeros.

Bit: 11
Name: Receiver Active (RCV ACT)
Function: This read-only bit is set at the center of the start bit of the serial input data and is cleared at the expected center of the stop bit at the end of the serial data.

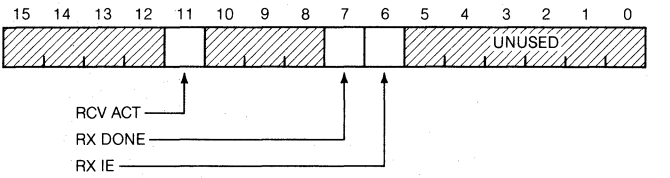


Figure 2-13 ▪ 17 777 560 Receiver Status Register

Bits: 10–8
Name: Unused. They always read as zeros.

Bit: 7
Name: Receiver Done (RX DONE)
Function: This read-only bit is set when an entire character has been received and is ready to be read from the RBUF register. This bit is automatically cleared when RBUF is read. It is also cleared by Powerup. RX DONE is set one bit time after RCV ACT clears.

Bit: 6
Name: Receiver Interrupt Enable (RX IE)
Function: This read-write bit is cleared by Powerup and bus INIT. If both RCVR DONE and RCVR INT ENB are set, a program interrupt is requested at priority level 4.

Bits: 5–0
Name: Unused. They always read as zeros.

- RECEIVER DATA BUFFER (RBUF)

Bit: 15
Name: Error (ERR)
Function: This read-only bit is set if RBUF bit 14 or bit 13 is set. ERR is clear if these two bits are clear.

Bit: 14
Name: Overrun Error (OVR ERR)
Function: This read-only bit is set if a previously received character was not read before being overwritten by the present character.

Bit: 13
Name: Framing Error (FRM ERR)
Function: This read-only bit is set if the present character had no valid stop bit. This bit is used to detect break if the front console panel switch is enabled and the break key is depressed on the console terminal.

NOTE

Error conditions remain present until the next character is received, at which point the error bits are updated. The error bits are not necessarily cleared by powerup.

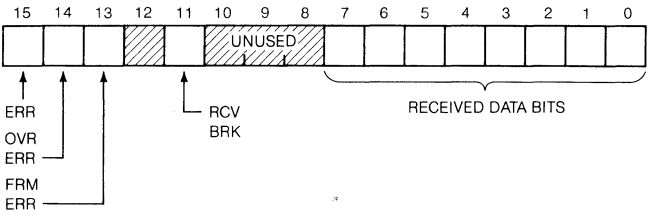


Figure 2-14 ▪ 17 777 562 Receiver Data Buffer

Bit: 12
Name: Unused. This bit always reads as zero.

Bit: 11
Name: Received Break (RCV BRK)
Function: This read-only bit is set at the end of a received character for which the serial data input remained in the SPACE condition for all 11 bit times. RCV BRK then remains set until the serial data input returns to the MARK condition.

Bits: 10–8
Name: Unused. These bits always read as zeros.

Bits: 7–0
Name: Received Data Bits
Function: These read-only bits contain the last received character.

▪ TRANSMITTER STATUS REGISTER (XCSR)

Bits: 15–8
Name: Unused. Read as zeros.

Bit: 07
Name: Transmitter Ready (TX RDY)
Function: This read-only bit is cleared when XBUF is loaded, and set when XBUF can receive another character. TX RDY is set by Powerup and by bus INIT.

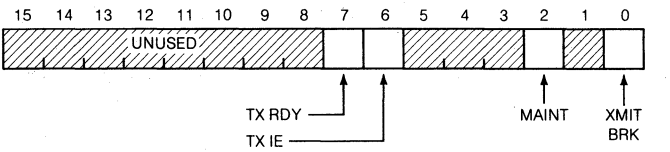


Figure 2-15 ▪ 17 777 564 Transmitter Status Register

Bit: 06
Name: Transmitter Interrupt Enable (TX IE)
Function: This read-write bit is cleared by Powerup and by bus INIT. If both TX RDY and TX IE are set, a program interrupt is requested.

Bits: 5–3
Name: Unused. Read as zeros.

Bit: 2
Name: Maintenance (MAINT)
Function: This read-write bit is used to facilitate a maintenance self-test. When MAINT is set, the DLART internally connects its output to its input. Input from the EIA receivers is ignored. The EIA output drivers are still connected and any characters sent with MAINT set will wrap around to the input and also go to the output drivers. This bit is cleared by Powerup and by bus INIT.

Bit: 1
Name: Unused. Read as zero.

Bit: 0
Name: Transmit Break (XMIT BRK)
Function: When this read-write bit is set, the serial output is forced to the SPACE condition. XMIT BRK is cleared by Powerup and by bus INIT.

- TRANSMITTER DATA BUFFER REGISTER (XBUF)

Bits: 15–8
Name: Unused.

Bits: 7–0
Name: ASCII character. These eight bits are write-only bits used to load the character to be transmitted.

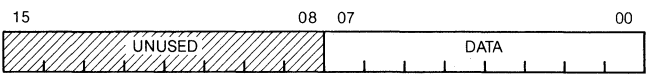


Figure 2-16 ▪ 17 777 566 Transmitter Data Buffer

• Line-time Clock

The Line Clock provides the system with timing information at fixed intervals determined by the UNIBUS line-time clock (LTC) signal from the power supply at 50 or 60 Hz, or by one of the on-board J-11 processor frequency signals as programmed by the boot and diagnostic controller status register bits 11 and 10. The three on-board frequencies are 50 Hz, 60 Hz and 800 Hz.

Clock Status Register (LKS)

The Clock Status Register (LKS) allows Line Clock interrupts to be enabled and disabled under program control.

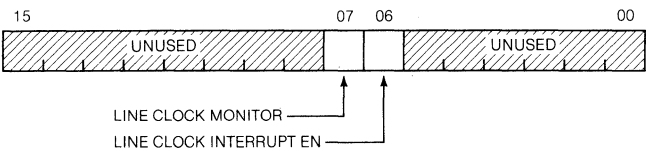


Figure 2-17 • 17 777 546 Clock Status Register

Bits: 15–8
Name: Unused. Always read as zeros.

Bit: 7
Name: Line Clock Monitor. ((LCM)
Function: This read-only bit is set by the leading edge of the external BEVENT line (or one of the three on-board frequencies) and by Bus INIT. Line Clock Monitor is cleared automatically on processor interrupts acknowledge. It is also cleared by writes to the LKS with bit 7 = zero.

Bit: 6
Name: Line Clock Interrupt Enable (LCIE)
Function: This read-write bit, when set, causes the set condition of LCM (LKS 7) to initiate a program interrupt request. When LCIE is clear, line clock interrupts are disabled. LCIE is cleared by Powerup and by Bus INIT. LCIE is held set when BCSR 13 (FRC LCIE) is set.

Bits: 5–0
Name: Unused. Always read as zeros.

▪ Console

The console terminal of the PDP-11/84 provides three functions.

- Setup of the system hardware
- Communication with the operating software
- Debugging of hardware and software

While using the console terminal to setup the system hardware, it is said to be in setup mode. While the console terminal is communicating with the operating software, it is said to be in program mode. While the PDP-11/84 is stopped, the console terminal is said to be in console-ODT mode.

The three modes are summarized in Table 2-4. Various conditions can switch the console terminal between modes. These conditions are diagrammed in Figure 2-18.

Table 2-4 ▪ PDP-11/84 Console Terminal Modes

	Setup-mode	Program-mode	Console-mode
Microprogram running?	Yes	Yes	Yes
Macroprogram running?	from ROM	from memory	No
UNIBUS functioning?	No	Yes	Yes

Setup Mode Functions

All setup of the PDP-11/84 CPU, except for the selection of the console baud rate, is performed via the console terminal using console-mode commands. Setup mode is a ROM-based program that can be entered when the ROM code is in dialog mode. Dialog mode allows the user to enter setup mode, map the memory and the I/O page, run ROM-based diagnostics, list available boot programs, and boot a device.

This setup dialog allows the user to determine the actions taken by the ROM code at powerup or restart including

- Select one of four powerup or restart modes
  - Enter dialog mode after running tests.
  - Autoboot selected devices after running tests.
  - Enter ODT HALT.
  - Restart operating system through powerup vector 24 for systems with battery backup memory.

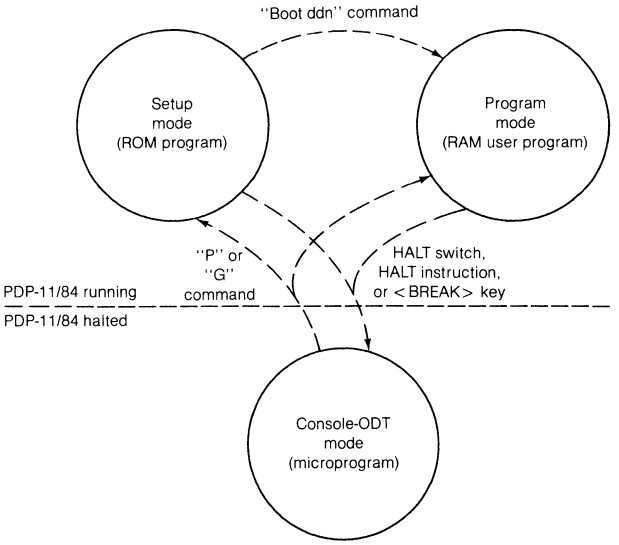


Figure 2-18 ▪ Conditions that Switch Console Terminal Modes

- Select register values and other parameters
  - Enable/disable the UNIBUS cache.
  - Enable/disable the line clock CSR and select clock source.
  - Select testing.
  - Enable/disable trap on HALT.
  - Override errors for nonstandard boot blocks.
  - Additional parameters. . .
- Bootstrap device selection for autoboot mode
  - Select one to six devices to try to boot.
  - Identify nonstandard CSR addresses.
- Support of custom bootstraps loaded in the EEROM
  - Create a custom bootstrap.
  - Delete a custom bootstrap.
  - Edit a custom bootstrap.

The setup dialogue appears as a series of menu pages that are compatible with hard- and soft-copy terminals.

- LEAVING SETUP MODE

Setup mode is exited by typing CTRL C. Then the user can either boot the operating system manually (i.e., BOOT DU0<CR>) or toggle the restart switch if autoboot mode is selected.

Program Mode Functions

While the console terminal is operating in program mode, it is under the sole control of the operating system software, and most of its functions are defined by that software.

- LEAVING PROGRAM MODE

The user enters console-ODT in one of three ways—by pressing the <BREAK> key while the PDP-11/84 keyswitch is in the <ENABLE> position, by placing the HALT/RUN/RESTART switch in the HALT position (again, while the keyswitch is in the <ENABLE> position), or by the PDP-11 program executing a HALT instruction while in kernel mode.

The user enters setup mode by simply pressing the HALT/RUN/RESTART switch to the RESTART position and typing CTRL C after the "Testing in progress—Please wait" message is displayed. Setup mode can be entered by setting the force dialog mode switch ON and pressing the HALT/RUN/RESTART switch to the restart position.

Console ODT

Besides machine setup, the console mode can also be used to examine or modify the contents of memory or UNIBUS device registers. Programs may also be stopped, started, or single-stepped. These functions are implemented by a microprogram contained within the J-11 processor chip. During the execution of these functions, the PDP-11/84 CPU is stopped. That is, although the J-11 microprogram is still running, no PDP-11 macroinstructions are being executed.

These functions are collectively refered to as console ODT (octal debugging technique) and are generally compatible with both ODT-11 and the console ODT functions provided by the LSI-11, the MicroPDP-11/23, and the PDP-11/24.

Console ODT can examine any read-only or read/write address in the entire 22-bit address space. Console ODT can also modify any read/write (not write-only) location. Locations which read all zeros (such as the console XBUF) can also be modified. The ODT commands are summarized in Table 2-5.

Table 2-5 • Console ODT Commands

Command	Symbol	Function
Slash	n/	Opens the specified location (n) and outputs its contents. The n is an octal number.
Carriage Return	<CR>	Closes an open location
Line Feed	<LF>	Closes an open location and then opens the next contiguous location.
Internal Register Designator	$n or Rn	Opens a specific processor register (n). Here n is an integer from 0 to 7 or the character S.
Processor Status Word Designator	S	Opens the PSW— must follow an $ or R command.
Go	G	Starts program execution.
Proceed	P	Resumes execution of a program.
Binary Dump	Control-S	Manufacturing use only

- LEAVING CONSOLE ODT MODE

The user can leave console-ODT mode via:

- The P command
- The G command
- The boot switch

If the user were previously in setup mode, the P or G command will exit you back to setup mode.

NOTE

The G command is used for a total restart, while P allows the user to proceed. The G command reinitializes the UNIBUS and resets memory.

If the user were previously in program mode, the P or G command will exit you back to program mode. The bootswitch's action depends on how you have setup the PDP-11/84.

▪ Specifications

Packaging

The basic PDP-11/84 is packaged as a 10.5-in by 19-in rackmount box product or a 19-in by 42-in cabinet product.

Standard Equipment

- PDP-11/84 CPU.
- Memory management.
- Line-frequency clock.
- 8-Kbyte cache memory.
- 1-Mbyte or 2-Mbyte ECC MOS memory.
- Floating-point accelerator (FPA) processor.
- Console serial line unit.
- Bootstrap and diagnostic ROM facility.
- Electrically erasable programmable ROMs (EEPROM).
- UNIBUS adapter (UBA) module.
- Monitor and distribution module.

Prewired Expansion Space for Optional Equipment

- 8 SPC slots for UNIBUS compatible peripherals, 7 hex and 1 quad.
- 1-Mbyte or 2-Mbyte ECC MOS memory (up to 4 Mbytes maximum).
- 0 SU open spaces in the box product.
- 2 SU open spaces in the cabinet product.

▪ Other Specifications

AC Power

Box	90–132 Vrms, 47–63 Hz, 1 phase power, 8 amp rms maximum @ 120 Vac 180–264 Vrms, 47–63 Hz, 1 phase power, 4.2 amp rms maximum @ 240 Vac
Cabinet	93–132 Vrms, 47–63 Hz, 1 phase power, 11.3 amp rms maximum @ 120 Vac 186–264 Vrms, 47–63 Hz, 1 phase power, 5.6 amp rms maximum @ 240 Vac

Physical Characteristics

Box	48.2 cm wide by 68.6 cm deep by 26.7 cm high (19 in by 27 in by 10.44 in)
Cabinet	53.9 cm wide by 80 cm deep by 105.7 cm high (21.25 in by 31.52 in by 41.64 in)

Weight

Box	44.5 kg (98 lb)
Cabinet	150 kg (331 lb)

Operating Environment

- BOX

Temperature:	10°C to 50°C (50°F to 122°F)
Humidity:	10% to 95% with maximum wet bulb of 32°C(90°F) (noncondensing)
Altitude:	To 2.4 km (8,000 ft)

- CABINET

Temperature	10°C to 40°C(50°F to 104°F)
Humidity:	10% to 90% with maximum wet bulb of 28°C(82°F) (noncondensing)
Altitude:	To 2.4 km (8,000 ft)

Nonoperating Environment

- BOX

Temperature:	−40°C to 66°C(−40°F to 151°F)

- CABINET

Temperature:	−40°C to 66°C(−40°F to 151°F)

The PDP-11/44 processor delivers the capability needed to satisfy a wide range of application requirements. Many outstanding features such as a high-performance central processor, access up to four Mbytes of main memory, and a large 8-Kbyte parity cache memory are standard on the PDP-11/44. Available options include the floating-point processor, commercial instruction set processor and the battery backup unit.

Many of the hardware features and expansion capabilites of the PDP-11/44 are common to other PDP-11s. Table 1-1 illustrates the similarities and differences between the PDP-11/44, PDP-11/84, PDP-11/70, PDP-11/24, and PDP-11/34A.

▪ System Architecture

The PDP-11/44 is a medium-scale, general purpose computer that is designed according to an enhanced, upwardly compatible version of the basic PDP-11 architecture. A block diagram is shown in Figure 3-1.

Memory management is standard with the basic computer, allowing expanded memory addressing, relocation, and protection. Also standard is a UNIBUS map that translates 18-bit UNIBUS addresses to 22-bit physical memory addresses. The cache contains 8,192 bytes of fast, static MOS memory that buffers the processor data from main memory.

The PDP-11/44 system has an expanded internal implementation of the PDP-11 architecture for greatly improved system throughput. All memory is on its own high-data-rate bus. The processor has a direct connection to the cache memory system for very rapid memory access.

The UNIBUS remains the primary control path in the PDP-11/44 system. It is conceptually identical with all other PDP-11 systems; the memory in the system still appears to be on the UNIBUS to all UNIBUS devices through the UNIBUS map. This expanded internal implementation of the PDP-11 architecture is generally compatible with earlier PDP-11/70 programs.

▪ Central Processor

The PDP-11/44 processor acts as the arbitration unit for UNIBUS control by regulating bus requests and transferring control of the bus to the requesting device with the highest priority.

The central processor contains arithmetic and control logic for a wide range of operations. These include fixed point arithmetic with hardware multiply and divide, extensive test and branch operations, and other control operations. It also provides room for the addition of the floating-point processor, commercial instruction set, and UNIBUS options.

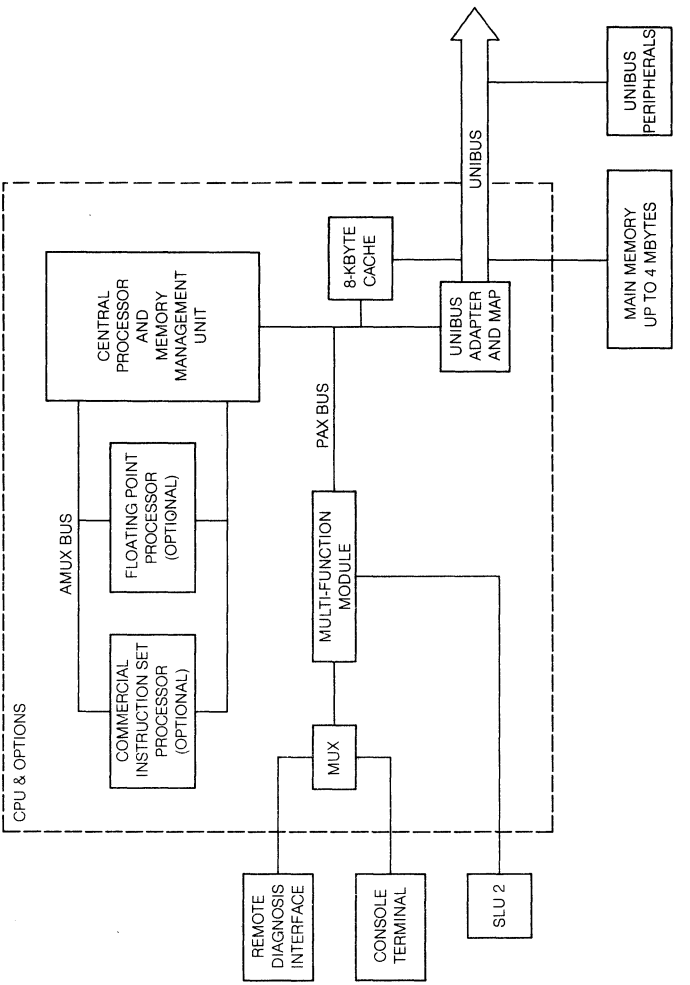


Figure 3-1 ▪ PDP-11/44 Block Diagram

The machine operates in three modes—kernel, supervisor, and user. When the machine is in kernel mode, a program has complete control of the machine; when the machine is in any other mode, the processor is inhibited from executing certain instructions and can deny direct access to the peripherals on the system. This hardware feature can be used to provide complete executive protection in a multiprogramming environment.

The central processor contains six general registers that can be used as accumulators, index registers, or stack pointers. Stacks are extremely useful for nesting programs, creating reentrant coding, and as temporary storage where a last-in/first-out structure is desirable. An additional register is used as the PDP-11/44's program counter. Three other additional registers are used as processor stack pointers, one for each operational mode.

The CPU performs all of the computer's computation and logic operations in a parallel binary mode through step-by-step execution of individual instructions.

General Registers

The general registers, as shown in Figure 3-2, can be used in many ways, the uses varying with requirements. The general registers can be used as accumulators, index registers, autoincrement registers, autodecrement registers, or as stack pointers for temporary storage of data. The *PDP-11 Architecture Handbook* chapter on Addressing Modes describes these uses of the general registers in more detail. Arithmetic operations can be done from one general register to another, from one memory location or device register to another, or between memory or a device register and a general register.

R7 is used as the machine's program counter (PC) and contains the address of the next instruction to be executed. It is a general register normally used for addressing purposes and not as an accumulator for arithmetic operations.

The R6 register is normally used as the processor stack pointer indicating the last entry on the current mode's hardware stack. (For information on the programming uses of stacks, please refer to the *PDP-11 Architecture Handbook*.) The three stacks are called the kernel stack, the supervisor stack, and the user stack. When the central processor is operating in kernel mode, it uses the kernel stack; in supervisor mode, the supervisor stack; and in user mode, the user stack. When an interrupt or trap occurs, the PDP-11/44 automatically saves its current status on the stack selected by the service routine. This stack-based architecture facilitates reentrant programming. The remaining six registers are R0-R5.

Registers can be used to increase the speed of realtime data handling or facilitate multiprogramming. Each of the six general registers could be used as an accumulator or index register for a realtime task or device.

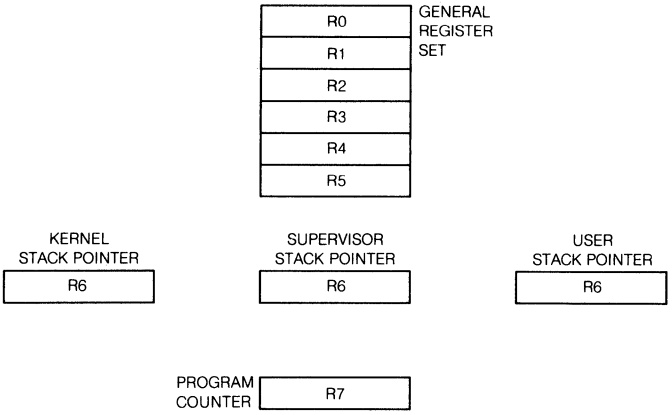


Figure 3-2 ▪ The General Registers

Processor Status Word (PSW)

The processor status word (PSW), shown in Figure 3-3, contains information on the current status of the PDP-11. This information includes current and previous operational modes, an indicator that is used to show that a Commercial Instruction Set (CIS) instruction was suspended by an interrupt, current processor priority, an indicator for detecting the execution of an instruction to be trapped during program debugging, and condition codes describing the results of the last instruction.

- MODES

Mode information includes the present mode, either user, supervisor, or kernel (bits 15, 14), and the mode the machine was in before the last interrupt or trap (bits 13, 12).

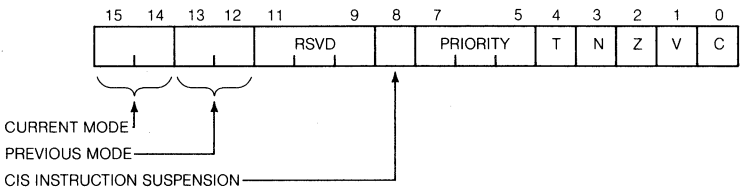


Figure 3-3 ▪ 17 777 776 Processor Status Word

The three modes permit a fully protected environment for a multiprogramming system by providing the user with three distinct sets of processor stacks and memory management registers for memory mapping.

In user and supervisor modes, a program is inhibited from executing a HALT instruction, and the processor will trap through location 4 if an attempt is made to execute this instruction. Furthermore, the processor will ignore the RESET and SPL (Set Priority Level) instructions, and will execute No Operation. In kernel mode, the processor will execute all instructions.

A program operating in kernel mode can map users' programs anywhere in memory and thus explicitly protect key areas (including the device registers and the processor status word) from the user operating environment.

- CIS INSTRUCTION SUSPENSION

  Bit 8, when set, indicates that a commercial instruction is in process. Because commercial instructions can be suspended (interrupted), this bit will be pushed onto the stack with the rest of the processor status word so that when control is returned to the routine, the commercial instruction can continue where it left off. Bit 8 may be used in future nonCIS instructions.

- PROCESSOR PRIORITY

  The central processor operates at any of eight levels of priority, 0-7. When the CPU is operating at level 7, an external device cannot interrupt it with a request for service. The central processor must be operating at a lower priority than the priority of the external device's request in order for the interruption to take effect. The current priority is maintained in the processor status word (bits 5-7). The eight processor levels provide an effective interrupt mask, which can be dynamically altered by the kernel-mode program through use of the SPL instructions. (For more information on the instructions see the *PDP-11 Architecture Handbook*.) This SPL instruction allows a kernel mode program to alter the central processor's priority without affecting the rest of the processor statu word.

- TRACE TRAP

  The trace trap bit (T) can be set or cleared under program control. When the trace trap bit is set, a processor trap will occur through location 14 on completion of instruction execution and a new processor status word and program counter will be loaded. This bit is especially useful for debugging programs because it provides an efficient method of single-stepping the program.

  Interrupt and trap instructions both automatically cause the previous processor status word and program counter to be saved and replaced by the new values corresponding to those required by the routine servicing the interrupt or trap.

The user can thus cause the central processor to automatically switch modes (context switching), alter the CPU's priority, or disable the trace trap bit whenever a trap or interrupt occurs.

- CONDITION CODES

  The condition codes contain information on the result of the last CPU operation. They include a negative bit (N), set if the result of the previous operation was negative; a zero bit (Z), set if the result of the previous operation was zero; an overflow bit (V), set if the result of the previous operation caused an arithmetic overflow, and a carry bit (C) set by the previous operation if the operation caused a carry out of its most significant bit.

- STACK LIMIT

  The PDP-11/44 has a kernel stack overflow boundary at location 400. Once the kernel stack exceeds this boundary, the processor will complete the current instruction and then trap through location 4, stack overflow in the CPU error register.

CPU Registers

The following CPU registers are accessed by program or console control.

CPU Error Register

The CPU error register (shown in Figure 3-4) identifies the source of the abort or trap that caused a trap through the vector at location 4. Bits 7–4, bit 2 and bit 0 are cleared when the CPU error register is written. When set, bit 9 indicates to software that a software powerdown is in progress. The remaining bits are software transparent and are accessible only when the console has control. They serve to provide diagnostic visibility into the processor.

Bit: 15
Name: Data Transfer
Function: Monitors the data transfer line of the processor. When clear, this bit indicates the processor is initiating a data transfer on the UNIBUS.

Bit: 14
Name: C1
Function: Set when the control signal Bus C1 is asserted, indicating that a DATO or DATOB transfer is being performed.

Bit: 13
Name: Cache Restart
Function: Set when the cache has generated the signal necessary to restart the processor clock.

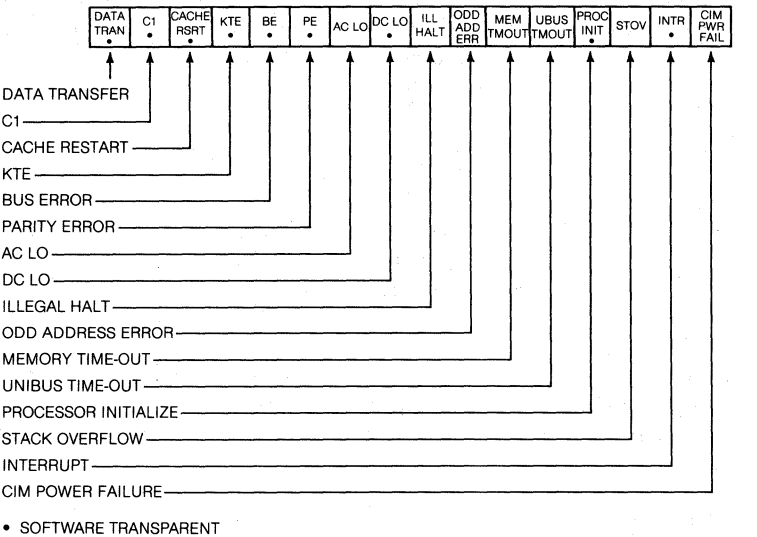


Figure 3-4 ▪ 17 777 766 CPU Error Register

Bit: 12
Name: KTE
Function: Set when a memory management error (nonresident, page length, or read-only abort) has occurred.

Bit: 11
Name: Bus Error
Function: Set when processor has attempted to access nonexistent memory, odd address during word reference, or if there was no response on the UNIBUS within approximately 20 microseconds.

Bit: 10
Name: Parity Error
Function: Set when processor has received an indication of a memory parity error.

Bit: 9
Name: AC LO
Function: Set when UNIBUS AC LO is asserted. To software, when this bit is set, a powerdown is in progress. This signal is not latched and therefore bit 9 is not affected by a processor INIT.

Bit: 8
Name: DC LO
Function: Set when UNIBUS DC LO is asserted. This signal is not latched and therefore bit 8 is not affected by a processor INIT.

Bit: 7
Name: Illegal Halt
Function: Set when a HALT instruction was attempted while the processor was in user or supervisor mode.

Bit: 6
Name: Odd Address Error
Function: Set when the program attempts a word reference of an odd address.

Bit: 5
Name: Memory Timeout
Function: Set when the processor attempts to read/write from a non-existent main memory location. This does not include UNIBUS addresses.

Bit: 4
Name: UNIBUS Timeout
Function: Set when the processor attempts to read/write from a nonexistent UNIBUS location.

Bit: 3
Name: Processor Initialize
Function: Set when the processor initialize signal is asserted.

Bit: 2
Name: Stack Overflow
Function: Set when the kernel hardware stack is below virtual address 400 octal.

Bit: 1
Name: Interrupt
Function: Set when the PAX interrupt line is asserted.

Bit: 0
Name: CIM Power Failure
Function: Set after dc power to the machine has exceeded voltage tolerance limits for a period of 1.5 microseconds or greater.

Processor Traps
Processor traps are a series of errors and programming conditions that will cause the central processor to trap through a set of fixed locations. These include power failure, odd addressing errors, stack errors, timeout errors, nonexistent memory references, parity errors, memory management violations, floating-point processor exception traps, use of reserved instructions, use of the T bit in the processor status word, and use of the IOT, EMT, and TRAP instructions.

- POWER FAILURE

Whenever ac power drops below 90 volts for 120 V power (180 volts for 240 V) or outside a limit of 47 to 63 Hz, as measured by dc power, the powerfail

sequence is initiated. The central processor automatically traps through location 24 and the user's powerfail program has 2 milliseconds to save all volatile information (data in registers).

If battery backup is present, and the batteries are not depleted when power is restored, the processor traps again through location 24 and executes the user's powerup routine to restore the machine to its state prior to power failure. If batteries are not present, a bootstrap of the default device is executed.

- ODD ADDRESSING ERROR

  This odd addressing error occurs whenever a program attempts to execute a word instruction on an odd address (between word boundaries). The instruction is aborted and the CPU traps through location 4.

- TIMEOUT ERROR

  This timeout error occurs when a MSYN pulse is placed on the UNIBUS or main memory bus and there is no SSYN pulse within 20 microseconds. This error usually occurs during attempts to address nonexistent memory or peripherals. The instruction is aborted and the processor traps through location 4.

- RESERVED INSTRUCTION

  There is a set of illegal and reserved instructions that causes the processor to trap through location 10. (e.g., if no floating-point processor is installed in the PDP-11/44, execution of a floating-point instruction results in a trap through location 10.)

- TRAP HANDLING

  The *PDP-11 Architecture Handbook* includes a list of the reserved trap vector locations and system error definitions that cause processor traps. When a trap occurs, the processor follows the same procedure for traps as it does for interrupts (saving the PC and PSW on the new processor stack, for example).

  In cases in which traps and interrupts occur concurrently, the processor will service the conditions according to the priority sequence following.

  1. HALT (instruction, switch, or command)
  2. Memory management fault
  3. Memory parity errors
  4. Bus error traps
  5. Floating-point traps
  6. TRAP instruction
  7. Trace trap
  8. Stack overflow trap
  9. Powerfail trap

10. Console bus request (console NEXT command or on-the-fly EXAMINE)
11. Program interrupt request (PIR) level 7
12. Bus request (BR) Level 7
13. PIR 6
14. BR 6
15. PIR 5
16. BR 5
17. PIR 4
18. BR 4
19. PIR 3
20. PIR 2
21. PIR 1
22. WAIT LOOP

- STACK LIMIT VIOLATIONS
  When instructions cause the kernel stack virtual address to go lower than 400 octal, a stack violation occurs. When the operation that caused the stack violation is completed, then a bus error trap is effected (trap through 4). The error trap, which itself uses the stack, executes without causing an additional violation.

Program Interrupt Requests (PIR)

Figure 3-5 shows the layout of the program interrupt request register. A request is booked by setting one of bits 15–9 (for PIR 7–PIR 1) in the program interrupt register. The hardware sets Bits 7–5 and 3–1 to the encoded value of the highest PIR bit set. This program interrupt active (PIA) field should be used to set the processor level and also index through a table of interrupt vectors for the seven software priority levels.

When the PIR is granted, the processor will trap through location 240 and pick up the PC from 240 and the PSW from 242. It is the interrupt service routine's responsibility to queue requests within a priority level and to clear the PIR bit before the interrupt is dismissed.

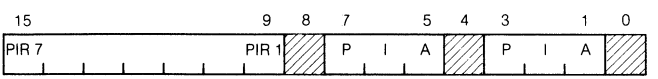


Figure 3-5 • 17 777 772 Program Interrupt Request Register

The following sample shows how the actual interrupt dispatch program should look.

```
MOVB PIR,PS        ;places bits 7–5 in PSW
                   :priority level bits
MOV R5,-(SP)       ;save R5 on the stack
MOV PIR,R5
BIC #177761,R5     ;gets bits 3–1
JMP @DISPAT(R5)    ;use to index through table
                   ;which requires 15 core
                   ;locations
```

CPU and I/O Device Registers and Addresses

The following, Table 3-1, summarizes the PDP-11/44 registers and their addresses.

Table 3-1 ▪ PDP-11/44 CPU and I/O Device Registers and Addresses

Address	Register
17 777 776	Processor Status Word (PSW)
17 777 772	Program Interrupt Request (PIRQ)
17 777 766	CPU Error
17 777 707—17 777 700	CPU General Registers
17 777 676—17 777 660	User Data PAR, Reg. 0-7
17 777 656—17 777 640	User Instruction PAR, Reg. 0-7
17 777 636—17 777 620	User Data PDR, Reg. 0-7
17 777 616—17 777 600	User Instruction PDR, Reg. 0-7
17 777 576	MM Status Register 2 (SR2)
17 777 574	MM Status Register 1 (SR1)
17 777 572	MM Status Register 0 (SR0)
17 777 570	Switch Register
17 777 566—17 777 560	Console Terminal SLU
17 777 776—17 760 000 (switch-selectable)	SLU 2 DECtape (Normally 17 776 500)
17 777 516	MM Status Register 3 (SR3)
17 772 376—17 772 360	Kernel Data PAR, Reg. 0-7
17 772 356—17 772 340	Kernel Instruction PAR, Reg. 0-7
17 772 336—17 772 320	Kernel Data PDR, Reg. 0-7
17 772 316—17 772 300	Kernel Instruction PDR, Reg. 0-7
17 772 276—17 772 260	Supervisor Data PAR, Reg. 0-7
17 772 256—17 772 240	Supervisor Instruction PAR, Reg. 0-7
17 772 236—17 772 220	Supervisor Data PDR, Reg. 0-7
17 772 216—17 772 200	Supervisor Instruction PDR, Reg. 0-7
17 770 372—17 770 200	UNIBUS Map Registers

Memory Systems

MOS Memory With Error Correcting Code and Optional Battery Backup

MOS memory with error correcting code (ECC) is identical to the PDP-11/84 system. For an explanation of ECC and the optional battery backup unit available for the PDP-11/44, refer to the PDP-11/84 chapter.

Memory Management

The Memory management hardware is standard with the PDP-11/44 computer. It is a hardware relocation and protection facility that can convert the 16-bit program virtual addresses to 22-bit physical addresses. The unit may be enabled or disabled under program control. There is a small speed advantage when the unit is in the 16-bit mode. For a more detailed description of memory management techniques, refer to the *PDP-11 Architecture Handbook*.

UNIBUS Map

The UNIBUS map is the hardware relocation facility for converting the 18-bit UNIBUS addresses to 22-bit addresses. The relocation mapping may be enabled or disabled under program control. Once again, there is a slight speed advantage when the UNIBUS map is disabled (off).

Cache Memory

PDP-11/44 cache memory is integral to the PDP-11/44 processor and is designed to increase the CPU performance by decreasing the CPU-to-memory read access time. It is an 8,192-byte, high-speed RAM memory, organized as a direct-mapped cache with write-through. Functionally, main memory and cache can be treated as a single unit (see Figure 3-1).

Physical Description

The PDP-11/44 cache memory interfaces to the processor through the processor backplane. Two user-accessible switches (S1 and S2) enable the cache to be shut off by causing a forced-miss condition in either upper or lower cache address space. Software bits for enabling or disabling cache are also provided in the processor's cache-control registers discussed later in this chapter.

General System Architecture

The cache operates as an associative memory in parallel with the main memory, and is connected to the CPU by the high-speed internal data path in the PDP-11/44 (the PAX data bus). This high-speed data path is separate from the internal data path that is shared by the floating-point and commercial instruction set options (the AMUX data bus). The cache is logically connected to the PAX address and memory address buses, but is isolated from them by a set of independent receivers. When a memory read transfer is initiated by the CPU,

the cache is strobed 100 nanoseconds later to determine if the data is in the cache and is error free. If so, this is referred to as a cache hit. If the access results in a cache hit, the processor clock is immediately restarted and clocks in the cache data that ends the transfer from the CPU. If access results in a cache miss, then main memory MSYN is asserted and the access is to main memory with the cache performing an automatic write-through to update itself. During write transfers, a write is performed to main memory with the cache updating itself if that location is presently cached. DMA write transfers from the UNIBUS are monitored by the cache and result in invalidation of cached locations. Only CPU transfers which access main memory are cached. Any data stored in memory appearing on the UNIBUS will not be cached.

CPU Bypass of the Cache

Besides having the capability of disabling half the cache, or the entire cache, the CPU can also disable caching of data based on the virtual address (virtual page) of the data. This is useful in two circumstances.

- If a multiported main memory (not supplied by Digital) is shared among two or more processors, it is possible for a particular word of main memory data to be cached in all of the CPUs. If one CPU then alters this word, only the copy of the data in main memory and that particular CPU's cache is updated. The other CPUs still have the old data in their caches. The old data is referred to as stale. In order to avoid using stale data, each CPU that accesses shared data must do so bypassing its cache. This ensures that the CPU gets the copy of the data stored in main memory, which is fresh (currently valid).
- Bypassing the PDP-11/44 cache is also useful if the CPU is sweeping through a large amount of data, with no intent of soon rereading the data. If the CPU is caching all the data, the cache will simply become full of useless data, meanwhile forcing out the program and other useful data. This is particularly true if the amount of data the CPU sweeps through exceeds 8,192 bytes. This technique of bypassing the cache while accessing large data lists does not apply to the PDP-11/70 or PDP-11/84.

Table 3-2 ▪ Cache Response Matrix

	CPU Hit	Miss	DMA Hit	Miss
Read Bypass	Nothing or Invalidate	Nothing	Nothing	Nothing
Write Bypass	Invalidate	Invalidate	Nothing	Nothing
Write	Update	Nothing	Invalidate	Nothing

The response of the cache to a CPU read bypass hit (see Table 3-2) is jumper selectable. In its normal configuration, jumper W1 (M7097 module) is

inserted and jumper W2 is removed to allow a forced miss to occur only for a CPU read hit bypass. If the PDP-11/44 and the KK11-B cache are to be used in a multiported memory system, jumper W1 is removed and jumper W2 is inserted to allow a CPU read hit with bypass to cause an invalidation to occur to that location. This allows the software to clear potentially stale cache data that might arise in a multiported memory system.

Cache Memory Organization

The cache memory array (Figure 3-6) consists of thirty 4096 X 1 RAM chips arranged as follows.

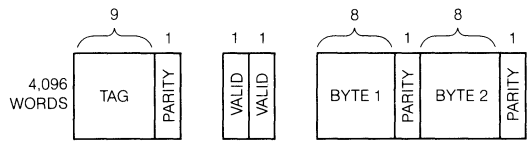


Figure 3-6 ▪ Cache Memory Array

TAG	Consists of nine tag store bits plus one bit of parity.
VALID	Consists of two bits, one of which is currently active, allowing the other bit to be cleared concurrently. By having two bits, a fast flush may be accomplished by switching to the set which has been previously cleared.
DATA	Consists of two 8-bit bytes plus a parity bit for each byte.

Cache-Control Registers

The following cache-control registers are implemented on the PDP-11/44 cache. All bits are cleared by processor INIT, but not by a CPU RESET instruction.

- CACHE DATA REGISTER (CDR)

Bits: 15–0 (Read Only)
Name: Cache Data Register Bits
Function: These bits are loaded from the 16-bit data array section of the cache RAM on every read access to main memory space, except the top 256 Kbytes,

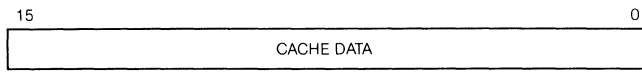


Figure 3-7 ▪ 17 777 754 Cache Data Register

which are reserved for the UNIBUS address space. This register can be used with the hit on destination only bit to aid the cache diagnostics in identifying failures in the data section of the cache array.

- CACHE MEMORY ERROR REGISTER (CMER)

Bit: 15
Name: Cache Memory Parity Error (CMPE)
Function: Set if a cache parity error is detected while the cache parity abort, bit 7, is set, or if a memory parity error occurs. If set, cache will force a miss. Clear by any write to the CMER or by console INIT. This bit must be cleared before the disable cache parity interrupt (DCPI) is cleared. If the cache detects a parity error in itself, the cache error LED (mounted on the cache module) will light.

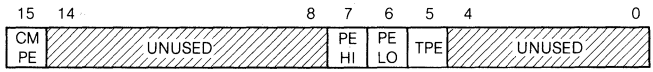


Figure 3-8 ▪ 17 777 744 Cache Memory Error Register

Bit: 7
Name: Parity Error High Byte (PEHI)

Bit: 6
Name: Parity Error Low Byte (PELO)

Bit: 5
Name: Tag Parity Error (TPE)
Function: These bits are set individually when a parity error occurs in the high-data byte, low-data byte, or tag field, respectively, if the cycle is aborted (cache parity abort bit is set). If the cycle is not aborted, all three bits, 5, 6 and 7, are set upon any cache parity error occurrence as an aid to system software compatibility. Cleared by any write to the CMPE register or by console INIT.

- CACHE CONTROL REGISTER (CCR)

Bit: 13 (Read Only)
Name: Valid Store in Use (VSIU)
Function: This bit indicates which set of valid store bits is currently being used to determine the validity of the contents of the tag store memory. It is complemented each time that the cache is flushed. When set, valid bit set B is in use. When clear, valid bit set A is in use.

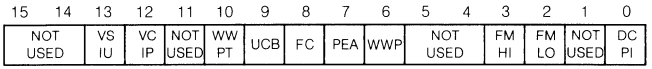


Figure 3-9 • 17 777 746 Cache Control Register

Bit: 12 (Read Only)
Name: Valid Clear in Progress (VCIP)
Function: This is set to indicate that the cache is currently in the process of clearing a valid store set. The clear cycle occurs on powerup and when the flush cache bit is set.

NOTE

The hardware clear cycle takes approximately 800 microseconds. While a valid store set is being cleared, the other set is in use allowing the cache to continue functioning. If the cache is flushed a second time within 800 microseconds, then the CPU will pause until the first flush completes (i.e., 800 microseconds from the time the first flush command was issued).

Bit: 10 (Read/Write)
Name: Write Wrong Tag Parity (WWTP)
Function: This bit when set causes tag parity bits to be written with wrong parity on CPU read misses and write hits. A parity error will thus occur on the next access to that location.

Bit: 9 (Read/Write)
Name: Unconditional Cache Bypass (UCB)
Function: When this bit is set, all references to memory by the CPU will be forced to go to main memory. Read or write hits will result in invalidation of those locations in the cache and misses will not change the contents.

Bit: 8 (Write Only)
Name: Flush Cache (FC)
Function: This bit will always read as zero. Writing a one into it will cause the entire contents of the cache to be declared invalid. Writing a zero into this bit will have no effect.

Bit: 7 (Read/Write)
Name: Parity Error Abort (PEA)
Function: This bit controls the response of the cache to a parity error. When set, a cache parity error will cause a forced miss and an abort to occur (asserts UNIBUS signal PB L). When cleared, this bit inhibits the abort and enables an interrupt through location 114. All cache parity errors result in forced misses.

Bit: 6 (Read/Write)
Name: Write Wrong Data Parity (WWDP)
Function: This bit when set causes high and low parity bytes to be written with wrong parity on all update cycles (CPU read misses and write hits). This will cause a cache parity error to occur on the next access to that location.

Bit: 3 (Read/Write)
Name: Force Miss High (FMHI)
Function: This bit when set causes forced misses to occur on CPU reads of addresses where physical address bit 12 is a one. This bit can also be set by moving the toggle switch S1 to the right side of the board. The bit cannot be cleared via the toggle switch.

Bit: 2 (Read/Write)
Name: Force Miss Low (FMLO)
Function: This bit when set causes forced misses to occur on CPU reads of addresses where physical address bit 12 is a zero. This bit can also be set by moving the toggle switch S2 to the right side of the board. The bit cannot be cleared via the toggle switch.

NOTE

Setting bits 3 and 2 will cause all CPU reads to be misses.

Bit: 0 (Read/Write)
Name: Disable Cache Parity Interrupt (DCPI)
Function: This bit when set overrides the cleared condition of the parity error abort bit, disabling the interrupt through location 114. The cache memory parity error bit must be cleared before disable cache parity interrupt (DCPI) is cleared.

Bit 7	Bit 0	Result of Cache Parity Error
0	0	Interrupt to 114 and force miss
0	1	Force miss only
1	X	Abort and force miss

▪ CACHE MAINTENANCE REGISTER (CMR)

Bits: 15–10 (Write Only)
Name: Address Match Bits 21–16
Function: This register is used to set bits 21–16 of the address-match register. The contents of the address-match register are constantly compared to the CPU physical address bus (PAX address). When an address-match occurs, the processor can 1) stop its microprogram; 2) halt; 3) supply an oscilloscope trigger pulse. This feature is useful for troubleshooting the PDP-11/44 system and is used with the console BREAK command.

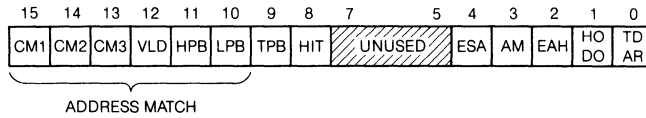


Figure 3-10 ▪ 17 777 750 Cache Maintenance Register

Bit: 15
Name: Compare 1 H

Bit: 14
Name: Compare 2 H

Bit: 13
Name: Compare 3 H

Bit: 12
Name: Valid H

Bit: 11
Name: High parity bit H

Bit: 10
Name: Low parity bit H

Bit: 9
Name: Tag parity bit H

Bit: 8
Name: Hit L
Function: These bits are key points in the cache that the diagnostic can use to help localize errors. This register is loaded on any read to main memory. Like the cache data register, these bits can be used with the hit on destination only bit to aid the cache diagnostic in tracing cache failure.

Bit: 4
Name: Enable Stop Action
Function: This bit can be set to allow the cache to stop the CPU clock upon detection of a cache parity error or address match condition. This stops the CPU microprogram.

Bit: 3 (Read/Write)
Name: Address Matched (AM)
Function: This bit is set when the 22-bit address match register is equal to the 22-bit cache address. The address-match LED (on the cache module) also lights.

Bit: 2 (Read/Write)
Name: Enable Halt Action
Function: This bit can be set to allow the cache to halt the CPU upon detection of a cache parity error or address match condition.

Bit: 1 (Read/Write)
Name: Hit on Destination Only (HODO)
Function: When set, this bit causes the cache to be enabled only during the final memory access of an instruction. Read hits and updates will happen only during the final access. This feature is a very powerful tool for cache diagnostics. When cleared, this bit has no effect on the cache. This bit should be used with caution because it can cause stale data in the cache.

Bit: 0 (Read/Write)
Name: Tag Data from Address Match Register (TDAR)
Function: When set, this bit enables the tag field of the cache to be written with data from bits 8–0 of the address match register. Once this bit is set, it will cause all cache writes to clear the valid bit in these locations. This feature allows the cache diagnostics to identify failures in the tag field of the cache array.

- CACHE HIT REGISTER (CHR)

Bits: 15–0 (Write Only)
Name: Address Match Bits
Function: This register is used to set bits 15–0 of the address match register. It is used in conjunction with bits 15–10 of the cache maintenance register.

Bits: 15–7 (Read Only)
Name: Tag Address Bits
Function: Tag address bits contain the nine bits of the tag store memory of the last access by the CPU to main memory (except the top 256 Kbytes). When used with the hit on destination only and tag data from address match register bits, this field will allow the cache diagnostics to read any tag field of any location in the array.

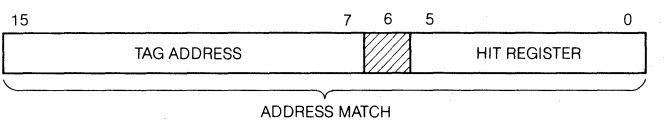


Figure 3-11 ▪ 17 777 752 Cache Hit Register (CHR)

Bits: 5–0 (Read Only)
Name: Hit Register
Function: This six-bit field shows the number of cache hits (read and write hits) on the last six CPU accesses to noncache-control memory. The bits flow from the least significant bit to the most significant bit of the field with a one indicating a hit and a zero indicating a miss.

• Other PDP-11/44 Processor Equipment

Floating-Point Processor

The PDP-11/44 floating point processor module fits integrally into the central processor. It provides a supplemental instruction set for performing single- and double-precision floating-point arithmetic operations and floating-integer conversion in parallel with the CPU. The floating-point processor provides both speed and accuracy in arithmetic computations. It provides 7 decimal digit accuracy in single-precision calculations and 17 decimal digit accuracy in double-precision calculations. For a detailed discussion on the PDP-11 floating-point processors, refer to the *PDP-11 Architecture HandBook*.

Backplane

Figure 3-12 illustrates the PDP-11/44 CA Backplane. In this diagram, the standard and optional hardware features are seen in their corresponding slots in the backplane.

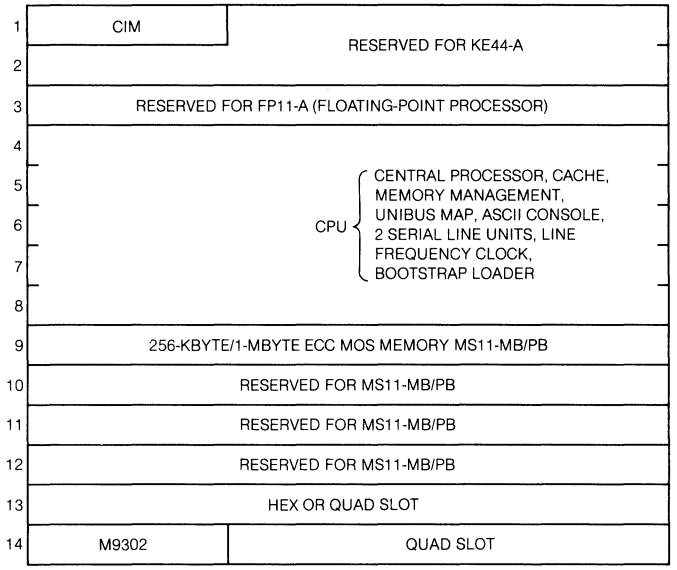


Figure 3-12 • PDP-11/44 Backplane Configuration

• Serial Line Unit Registers

The PDP-11/44 contains two serial line interfaces as a standard feature. The first interface (the console interface) is used to control the PDP-11/44 hardware and the operating system software.

While controlling the PDP-11/44 hardware, the console interface is said to be in console mode and is operated by a fixed program in both systems. While controlling, and in turn being controlled by, operating system software, the console interface is said to be in program mode. The selection of console or program mode is made via the front panel and by special characters typed at the console terminal.

A detailed description of the console ODT commands for the PDP-11/44 is found in Appendix B.

Serial Line Unit Timing Considerations

The UART (Universal Asynchronous Receiver/Transmitter) is an asynchronous subsystem. The transmitter accepts parallel characters and converts them to serial asynchronous output. The receiver accepts asynchronous serial characters and converts them to parallel output.

- RECEIVER

  The RECEIVER DONE bit sets when the UART has assembled a full character, which occurs approximately at the middle of the first stop bit. Because the UART is double-buffered, data remains valid until the next character is received and assembled. This allows one full character time for servicing the RECEIVER DONE bit or the interrupt caused by it.

- TRANSMITTER

  The UART's transmitter section is also double-buffered. After initialization, the TRANSMITTER READY bit is set. When the buffer is loaded with the first character, the bit clears but sets again within a fraction of a character transmission time period. A second character can then be loaded, clearing the bit again. This time the bit remains clear until the first character and its stop bit(s) have been transmitted (about one character time).

- BREAK GENERATION

  Setting the break bit causes the transmission of a continuous space. Because the TRANSMITTER READY bit continues to function normally, the duration of the break can be timed by the "pseudo-transmission" of a number of characters. However, because the transmitter is double-buffered, a null character (all zeros) should precede transmission of the break to ensure that the previous character completes transmission. Likewise, the last "pseudo-transmitted" character under break should be a null.

Terminal Serial Line Unit Control Registers (SLU 1)

There are four terminal SLU registers that follow. All unused or write only bits are zero when examined.

- RECEIVER CONTROL STATUS REGISTER (TERM RCSR)

Bits: 15–8
Function: Unused

Bit: 7 (Read Only)
Name: RECEIVER DONE
Function: Set during the program mode when an entire character has been received and is ready for transfer to the CPU. Cleared by INIT or by addressing (read only) RBUF. Starts an interrupt sequence when set if RECEIVER INTERRUPT ENABLE is also set.

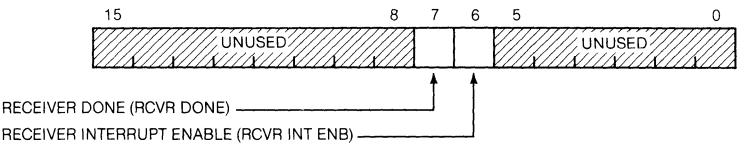


Figure 3-13 ▪ 17 777 560 Receiver Control Status Register

Bit: 6 (Read/Write)
Name: RECEIVER INTERRUPT ENABLE
Function: Cleared by INIT. When set, a priority 4 interrupt sequence will start each time RECEIVER DONE is set.

Bits: 5–0
Function: Unused

- RECEIVER DATA BUFFER (TERM RBUF)

Bit: 15 (Read Only)
Name: ERROR
Function: Logical OR of OVERRUN ERROR, FRAMING ERROR and PARITY ERROR. ERROR is not tied to the interrupt logic, but RECEIVER DONE is.

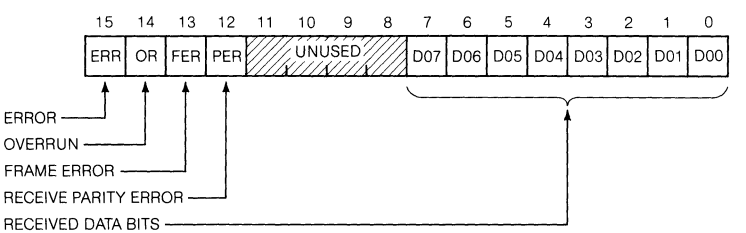


Figure 3-14 ▪ 17 777 562 Receiver Data Buffer

Bit: 14 (Read Only)
Name: OVERRUN ERROR
Function: Set if the previously received character is not read (RECEIVER DONE not cleared) before another character is received.

Bit: 13 (Read Only)
Name: FRAMING ERROR
Function: Set if the character received has no valid stop bit(s). Also used to detect a "break" character.

Bit: 12 (Read Only)
Name: PARITY ERROR
Function: Set if received parity does not agree with the expected parity. Always cleared if no parity is selected.

NOTE

Error bits remain set until the next character is received, at which time the error bits are updated. INIT does not clear the console terminal error bits. However, a powerup sequence does clear them. Error bits may be disabled by removing a jumper on the M7096 module.

Bits: 11–8
Function: Unused

Bits: 7–0 (Read Only)
Name: RECEIVED DATA
Function: These bits contain the character just received. If fewer than eight bits are selected, the buffer will be right-justified with the unused bits read as 0. Not cleared by INIT.

- TRANSMITTER CONTROL STATUS REGISTER (TERM XCSR)

Bits: 15–8
Function: Unused

Bit: 7 (Read Only)
Name: TRANSMITTER READY
Function: Set during the program mode only by INIT or when XBUF can accept another character. Cleared when a character is written into the XBUF. Starts an interrupt sequence if TRANSMITTER INTERRUPT ENABLE is also set.

Bit: 6 (Read/Write)
Name: TRANSMITTER INTERRUPT ENABLE
Function: Cleared by INIT. When set, a priority 4 interrupt sequence will start each time TRANSMITTER READY is set.

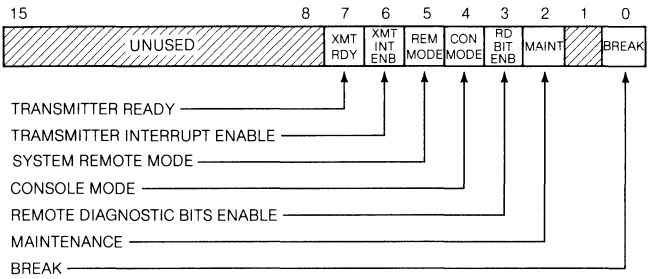


Figure 3-15 • 17 777 564 Transmitter Control Status Register

Bit: 5 (Read Only)
Name: SYSTEM REMOTE MODE
Function: Set when CPU is operating in the remote diagnostic mode.

Bit: 4 (Read Only)
Name: CONSOLE MODE
Function: Set to indicate that the CPU is operating in the console mode.

Bit: 3 (Read Only)
Name: REMOTE DIAGNOSTIC BITS ENABLE
Function: Set by turning on switch #2 of E79 on the M7096 module. When set, the statuses of bits 4 and 5 are entered into this register. When cleared (switch off), all three bits will be zero.

Bit: 2 (Read/Write)
Name: MAINTENANCE
Function: Cleared by INIT. When set, it connects the serial output of the TRANSMITTER into the serial input of the RECEIVER, in place of the normal serial input from the terminal. It also forces the receiver to run at the same speed as the transmitter.

Bit: 1
Function: Unused

Bit: 0 (Read/Write)
Name: BREAK
Function: Cleared by INIT. When set, a continuous space is transmitted, equivalent to sending a null character with no stop bits (framing error). May be disabled by removing a jumper on the M7096 module.

▪ TRANSMITTER DATA BUFFER (TERM XBUF) 17 777 566

Bits: 15–8
Function: Unused

Bits: 7–0 (Write Only)
Name: TRANSMITTER DATA BUFFER
Function: If fewer than eight bits are jumper selected, the character must be right-justified.

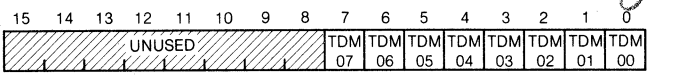


Figure 3-16. ▪ 17 777 566 Transmitter Data Buffer

Second Serial Line Unit Registers (SLU 2)
The second serial line unit is a general-purpose serial line interface. It may be used for a variety of purposes.

- Connection of a serial line-printer
- Connection of a TU58 cartridge tape drive
- Connection of a modem

This interface is not recomended for use with a high-speed (>1200 baud) interactive terminal, such as a VT220 or VT240/Z. The four SLU2 registers follow.

▪ RECEIVER CONTROL/STATUS REGISTER (SLU 2 RCSR)

Bits: 15–8
Function: Unused

Bit: 7 (Read Only)
Name: RECEIVER DONE
Function: Set when an entire character has been received and is ready for transfer to the CPU. Cleared by INIT or addressing (read-only) RBUF. Starts an interrupt sequence when set if RECEIVER INTERRUPT ENABLE is also set.

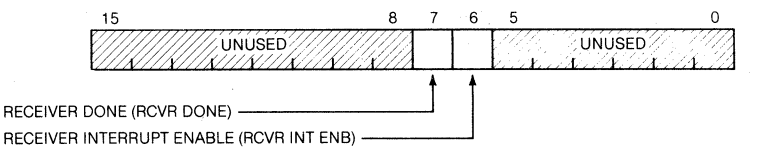


Figure 3-17 ▪ Receiver Control/Status Register

Bit: 6 (Read/Write)
Name: RECEIVER INTERRUPT ENABLE
Function: Cleared by INIT. When set, a priority 4 interrupt sequence will start each time RECEIVER DONE is set.

Bits: 5–0
Function: Unused

- RECEIVER DATA BUFFER (SLU 2 RBUF)

Bit: 15 (Read Only)
Name: ERROR
Function: Logical OR of OVERRUN ERROR, FRAMING ERROR and PARITY ERROR. ERROR is not tied to the interrupt logic, but RECEIVER DONE is cleared by INIT. Bits 12 through 15 may be disabled and cleared by removing a jumper on the M7096 module.

Bit: 14 (Read Only)
Name: OVERRUN ERROR
Function: Set if previously received character is not read (RECEIVER DONE not cleared) before another character is received. Cleared by INIT or reading before receiving another character.

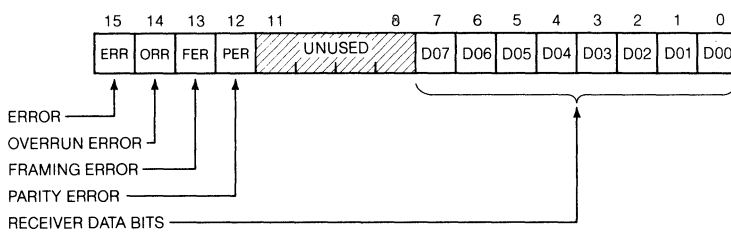


Figure 3-18 • Receiver Data Buffer

Bit: 13 (Read Only)
Name: FRAMING ERROR
Function: Set if character received has no valid stop bit(s). Cleared by INIT or when a valid character is received. This bit indicates an error in transmission or the reception of a "break" character.

Bit: 12 (Read Only)
Name: PARITY ERROR
Function: Set if received parity does not agree with expected parity. Cleared by INIT or when the parity of the next character is valid. Always cleared if no parity is selected.

Bits: 11–8
Function: Unused

Bits: 7–0 (Read Only)
Name: RECEIVED DATA
Function: These bits contain the character just received. If fewer than eight bits are selected, the buffer will be right-justified with the unused bits read as zero. Not cleared by INIT.

- TRANSMITTER CONTROL/STATUS REGISTER (SLU 2 XCSR)

Bits: 15–8
Function: Unused

Bit: 7 (Read Only)
Name: TRANSMITTER READY
Function: Set by INIT or when the XBUF can accept another character. Starts an interrupt sequence when set if TRANSMITTER INTERRUPT ENABLE is also set. Cleared when a character is written into the XBUF.

Bit: 6 (Read/Write)
Name: TRANSMITTER INTERRUPT ENABLE
Function: Cleared by INIT. When set, a priority 4 interrupt sequence will start each time TRANSMITTER READY is set. Cleared by the program or by the initialization sequence.

Bits: 5–3
Function: Unused

Bit: 2 (Read/Write)
Name: MAINTENANCE
Function: Cleared by INIT. When set, it connects the serial output of the transmitter into the serial input of the receiver, in place of the normal serial input from the terminal. It also forces the receiver to run at the same speed as the transmitter.

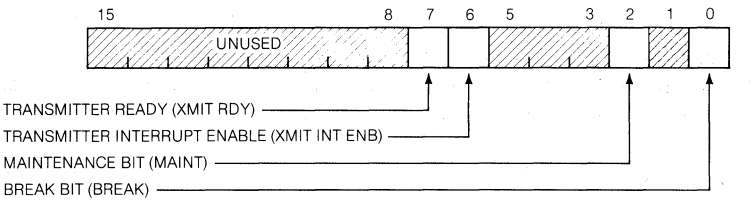


Figure 3-19 ▪ Transmitter Control/Status Register

Bit: 1
Function: Unused

Bit: 0 (Read/Write)
Name: BREAK
Function: Cleared by INIT. When set, a continuous space is transmittd equivalent to sending a null character with no stop bits (framing error). May be disabled by removing a jumper on the M7096 module.

- TRANSMITTER DATA BUFFER (SLU 2 XBUF)

Bits: 15–8
Function: Unused

Bits: 7–0 (Write Only)
Name: TRANSMITTER DATA
Function: If fewer than eight bits are selected, the character must be right-justified.

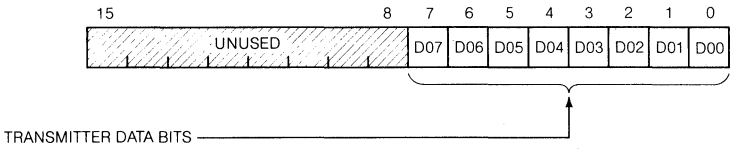


Figure 3-20 • Transmitter Data Buffer

• Line Clock

The PDP-11/44 includes a line-time clock as standard equipment. This clock provides interrupts synchronized with the cycles of the ac power line (mains). By counting these interrupts, this allows the operation system software to keep realtime.

Line Clock Status Register

Bits: 15–8
Function: Unused

Bit: 7 (Read/Write, clear only)
Name: LINE CLOCK MONITOR
Function: Set by INIT or by the line-frequency clock signal (LTC). Cleared only by the program.

Bit: 6 (Read/Write)
Name: LINE CLOCK INTERRUPT ENABLE
Function: Cleared by INIT. When set, starts a priority 4 interrupt sequence each time LINE CLOCK MONITOR is set.

Bits: 5–0
Function: Unused

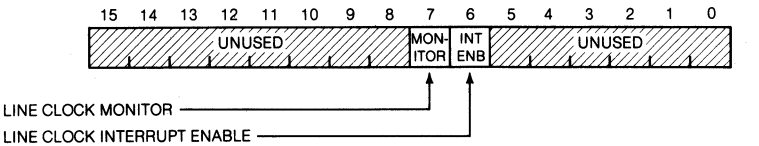


Figure 3-21 ▪ 17 777 546 Line Clock Status Register

▪ Address and Vector Assignments

Integral to the PDP-11/44 CPU are the two serial line units and the line clock. The serial line units and clock follow the same address and vector assignments as the KL11, DL11-A, B, C, D, and W, and the KW11-L, respectively. SLU #1 is used for the system console and has fixed addresses and vectors. SLU #2, which may be used for the TU58 or other asynchronous devices, has switch-selectable contiguous addresses and vectors. The realtime clock has a fixed address and vector.

Table 3-3 ▪ PDP-11/44 Address and Vector Assignments

	Address	Vector	Priority
Console	17 777 560		BR4 (fixed)
(SLU #1)	17 777 562	60	
	17 777 564		
	17 777 566	64	BR4 (fixed)
(SLU #2)	17 7YX XX0		BR4 (fixed)
	17 7YX XX2	XX0	
	17 7YX XX4		
	17 7YX XX6	XX4	BR4 (fixed)
	Where Y=6 or 7 and X=0-7 (vector)		
Line Clock	17 777 546	100	BR6 (fixed)

NOTE

Recommended address and vector assignments for SLU #2 when used for a TU58 are:
Address: 17776500
Vector: 300
These are the settings as received from Digital.

• Specifications

Packaging

A basic PDP-11/44 consists of a 10.5-in box with a 14-slot backplane, power supply, CPU, and 1-Mbyte memory.

Component Parts

The basic PDP-11/44 system includes

- Standard Equipment
  - -PDP-11/44 CPU
  - -Memory management.
  - -Bootstrap loader.
  - -Line-frequency clock.
  - -Asynchronous console terminal interface.
  - -Second asynchronous serial interface.
  - -8-Kbyte cache memory.
  - -1-Mbyte ECC MOS Memory.
  - -BA11-A box with power supply.

- Prewired Expansion Space for Optional Equipment
  - -Floating-point processor.
  - -Commercial instruction set processor.
  - -2 SPC slots for peripherals (1 hex, 1 quad).
  - -3-Mbyte ECC MOS memory (up to 4 Mbytes maximum).
  - -3 SU open space in CPU box.

Other Specifications

- AC POWER
  90-128 Vrms, 47-63 Hz, 1 phase power, 19 amp rms maximum @ 120 Vac
  180-256 Vrms, 47-63 Hz, 1 phase power, 9.5 amp rms maximum @ 240 Vac

- SIZE
  Each cabinet is 26.4 cm high by 42.2 cm wide by 66.0 cm deep (10.4-in by 16.6-in by 26.0-in)

- WEIGHT

CPU box	40.5 kg (90 lbs)

- OPERATING ENVIRONMENT

Temperature:	5°C to 50°C (41°F to 122°F)
Humidity:	10% to 95% with max. wet bulb of 32°C (89.6°F) and minimum dew point of 2°C (36°F)
Altitude:	To 2.4 km (8,000 ft.) noncondensing

- NONOPERATING ENVIRONMENT

Temperature:	-40°C to 80°C (-40°F to 176°F)
Humidity:	To 95% noncondensing
Altitude:	To 9.1 km (30,000 ft.)


RLO2

The PDP-11/24 is a fourth-generation member of the UNIBUS PDP-11 processor family. Designed as a single hex-module UNIBUS processor, the PDP-11/24 provides the basis for compact, low-cost application solutions. Offering an extended 22-bit memory addressing capability, the PDP-11/24 can address up to four full Mbytes of memory. The PDP-11/24 optional floating-point unit and commercial instruction set provide programming compatibility with other PDP-11s.

Integral to the PDP-11/24 central processor unit are hardware features and expansion capabilities that are common to the PDP-11/84, PDP-11/44, PDP-11/70 and PDP-11/34A. Table1-1 illustrates the similarities and differences between these five minicomputers.

▪ System Architecture

The PDP-11/24 is a minicomputer designed for both multitasking and dedicated applications. A block diagram of the computer is shown in Figure 4-1.

The central processor performs all arithmetic and logical operations required in the system. Memory management is standard with the basic computer, allowing expanded memory addressing, relocation, and protection. The UNIBUS map, which translates UNIBUS addresses to physical memory address, is program compatible with PDP-11/44 and PDP-11/84 UNIBUS maps. The UNIBUS remains the primary control path in the PDP-11/24 system. Memory addresses are passed on a separate 22-bit wide bus. This bus provides reduced memory access times. It is conceptually identical with previous PDP-11 systems; the memory in the system still appears to be on the UNIBUS to all UNIBUS devices through the UNIBUS map.

▪ Central Processor

The PDP-11/24 processor is the arbitration unit for UNIBUS control. It regulates bus requests and transfers control of the bus to the request device with the highest priority.

The central processor contains arithmetic and control logic for a wide range of operations. These include fixed-point arithmetic with hardware multiply and divide, extensive test and branch operations, and other control operations. It also provides room for the addition of the floating point unit, commercial instruction set, and UNIBUS options.

The machine operates in two modes—kernel and user. When the machine is in kernel mode, a program has complete control of the machine; when the machine is in user mode, the processor is inhibited from executing certain instructions and can be denied direct access to the peripherals on the system. This hardware feature can be used to provide complete executive protection in a multiprogramming environment.

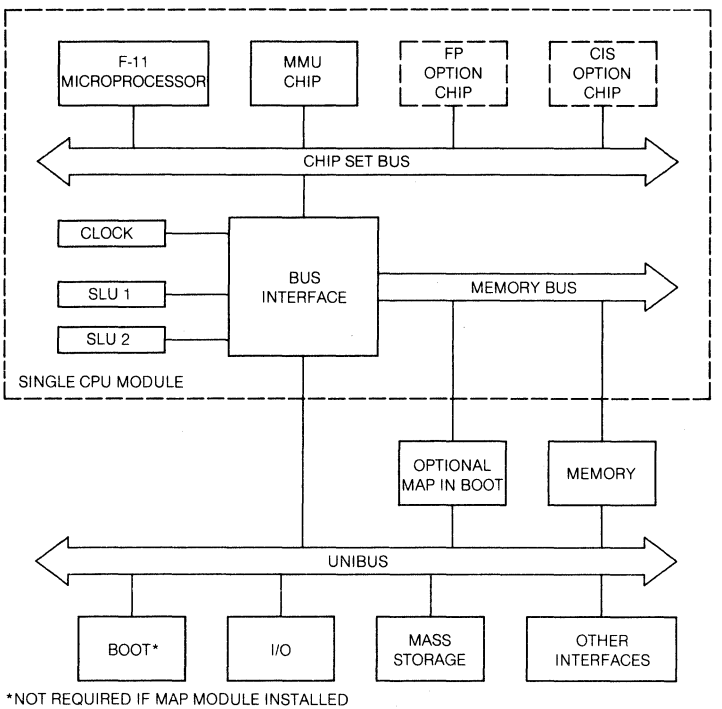


Figure 4-1 ▪ PDP-11/24 Block Diagram

The PDP-11/24 processor is implemented using three chips. Two MOS/LSI chips, called the data chip and control chip, implement the basic processor (see Figure 4-2). The memory management unit (MMU), the third chip, provides a software-compatible memory management scheme.

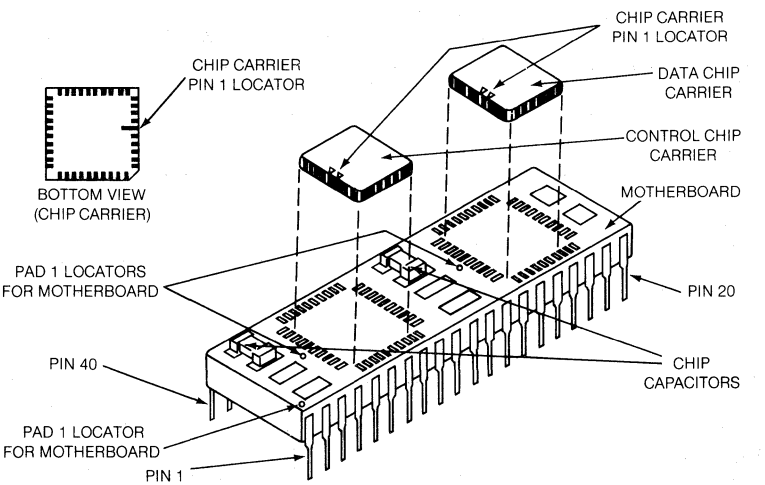


Figure 4-2 ▪ PDP-11/24 Data and Control Chip

The data chip (DC302) performs all arithmetic and logical functions, handles data and address transfers with the external world, and coordinates most interchip communication. The control chip (DC303) does microprogram sequencing for PDP-11 instruction decoding and contains the control store ROM. The data and control chips are both contained on one 40-pin package. The MMU chip (DC304) contains the registers for 18-bit or 22-bit memory addressing and also includes the FP11 floating-point registers and accumulators.

▪ Registers

The central processor contains nine registers that can be used as accumulators, index registers, or stack pointers for temporary storage of data. Six of these registers, R0-R5, are general registers (see Figure 4-3) that increase the speed of realtime data handling and facilitate multiprogramming. They can be used as accumulators or index registers for a realtime task or device. Another register,

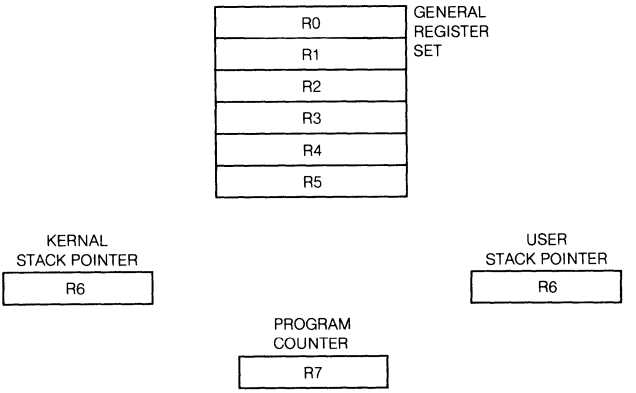


Figure 4-3 ▪ The General Registers

R7, is the PDP-11/24's program counter (PC). It is normally used for addressing purposes, not as an accumulator for arithmetic operations. This register contains the address of the next instruction to be executed. The other registers, R6, are processor stack pointers (SP). They maintain their respective hardware stacks, kernel and user. Additional registers are reserved for internal machine use. The *PDP-11 Architecture Book* describes the functions and operations of the registers in more detail.

Stacks are used for nesting programs, creating reentrant coding and temporary storage when a last-in/first-out structure is designed. For more information on programming uses of stacks, please refer to the *PDP-11 Architecture Handbook*.

Processor Status Word (PSW)

The processor status word (PSW) contains information on the current status of the PDP-11/24. This information includes current and previous operational modes, an indicator that is used to show that a CIS instruction was suspended by an interrupt, current processor priority, an indicator for detecting the execution of an instruction to be trapped during program debugging, and condition codes describing the results of the last instruction.

- MODES

  Mode information includes the present mode, either user or kernel (bits 15, 14), and the mode the machine was in before the last interrupt or trap (bits 13, 12).

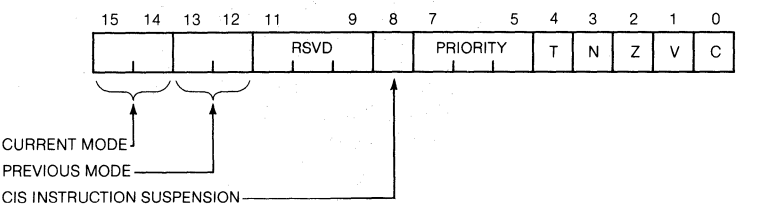


Figure 4-4 ▪ 17 777 776 Processor Status Word

The two modes permit a fully protected environment for a multiprogramming system by providing the user with two distinct sets of processor stacks and memory management registers for memory mapping.

When in user mode, a program is inhibited from executing a HALT instruction and the processor will trap through location 4 if an attempt is made to execute this instruction. Furthermore, the processor will ignore the RESET instruction, and execute No Operation. In kernel mode, the processor will execute all instructions.

A program operating in kernel mode can map users' programs anywhere in memory and thus explicitly protect key areas (including device registers and the PSW) from the user operating environment.

- CIS INSTRUCTION SUSPENSION

  When set, bit 8 indicates that a commercial instruction is in process. Since commercial instructions can be interrupted, this bit will be pushed onto the stack with the rest of the processor status word. When control is returned to the routine, the commercial instruction will continue where it left off.

- PROCESSOR PRIORITY

  The central processor operates at any of eight levels of priority, 0-7. When the CPU is operating at level 7, an external device cannot interrupt it with a request for service. The central processor must be operating at a lower priority than the priority of the external device's request in order for the interruption to take effect. The current priority is maintained in the PSW (bits 5-7). The eight processor levels provide an effective interrupt mask, which can be dynamically altered by the kernel-mode program through use of the set priority level instruction. (For more information on the instructions, see the *PDP-11 Architecture Handbook*).

- TRACE TRAP

  The trace trap bit (T) can be set or cleared under program control. When set, a processor trap will occur through location 14 after the execution of the

instruction is completed, and a new PSW will be loaded. This bit is especially useful for debugging programs because it provides an efficient method of single-stepping the program.

Interrupt and trap instructions both automatically cause the previous processor status word and program counter to be saved and replaced by the new values corresponding to those required by the routine servicing the interrupt or trap. The user can thus cause the central processor to automatically switch modes (context switching), alter the CPU's priority, or disable the trace trap bit whenever a trap or interrupt occurs.

- CONDITION CODES

The condition codes contain information on the result of the last CPU operation. They include a negative bit (N), set if the result of the previous operation was negative; a zero bit (Z), set if the result of the previous operation was zero; an overflow bit (V), set if the result of the previous operation caused an arithmetic overflow, and a carry bit (C) set by the previous operation if the operation caused a carry out of its most significant bit.

- STACK LIMIT

The PDP-11/24 has a kernel stack overflow boundary at location 400. Once the kernel stack goes below this boundary, the processor will complete the current instruction and then trap through location 4, indicating stack overflow in the CPU error register.

CPU Error Register

This register is available only when the UNIBUS map is installed. The CPU error register contains one bit, bit 0. This bit when set indicates that one or more power supply voltages has exceeded its tolerance. This bit is set when voltage error occurs and is cleared either by RESET or by writing a zero to the bit.

Bit: 0
Name: CPU Power Failure
Function: (See explanation above.)

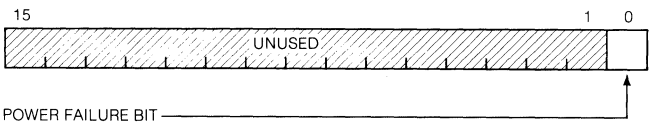


Figure 4-5 ▪ 17 777 766 CPU Error Register

Processor Traps

These are several kinds of errors and programming conditions that will cause the central processor to trap through a set of fixed locations. These include power failure, stack errors, timeout errors (nonexistent memory references), memory errors, memory management violations, floating-point processor exception traps, use of reserved instructions, use of the T bit in the PSW, and use of the IOT, EMT, BPT, and TRAP instructions.

- POWER FAILURE

  Whenever ac power drops below 90 volts for 120 V power (180 volts for 240 V) or outside a limit of 47 to 63 Hz, as measured by dc power, the powerfail sequence is initiated. The central processor automatically traps through location 24 and the user's powerfail program has approximately 5 milliseconds to save all volatile information (data in registers, I/O status, etc.) and to condition peripherals for power failure.

  If battery backup is present and if the batteries are not depleted when power is restored, the processor again traps to location 24 and executes the user's powerup routine to restore the machine to its state prior to power failure. If batteries are not present, a bootstrap of the default device is executed.

- TIMEOUT ERROR

  This timeout error occurs when a MSYN pulse is placed on the UNIBUS and there is no SSYN pulse within 20 microseconds. This error usually occurs in attempts to address nonexistent memory or peripherals. The instruction is aborted and the processor traps through location 4.

- RESERVED INSTRUCTION

  There is a set of illegal and reserved opcodes that causes the processor to trap through location 10. An example would be an attempt to execute a floating-point instruction when no floating-point processor is present.

- TRAP HANDLING

  The *PDP-11 Architecture Handbook* includes a list of the reserved trap vector locations and system error definitions that cause processor traps. When a trap occurs, the processor follows the same procedure for traps as it does for interrupts (saving the PC and PSW on the new Processor Stack, etc.). In cases in which multiple traps and interrupts occur concurrently, the processor will service the conditions according to the priority sequence that follows.

- TRAP PRIORITIES

1. DC LO (powerup)
2. Reserved instruction trap
3. Memory management fault
4. Bus error traps
5. Memory parity errors
6. Trace trap
7. Stack overflow trap
8. Powerfail trap
9. Bus request (BR) level 7
10. Line clock (highest B6 device)
11. BR level 6
12. BR level 5
13. BR level 4
14. HALT REQUEST
15. WAIT LOOP

- STACK LIMIT VIOLATIONS

When instructions cause the kernel R6 to exceed (go lower than) 400_8, a stack limit violation occurs. When operations that cause a stack limit violation are completed, then a bus error trap is effected (Trap to 4). The error trap, which itself uses the stack, executes without causing an additional violation.

CPU and I/O Device Registers and Addresses

Table 4-1 • PDP-11/24 CPU and I/O Device Registers and Addresses

Address	Register
17 777 776	Processor Status Word (PSW)
17 777 766	CPU Error (Optional with UNIBUS map)
17 777 707—17 777 700	CPU General Register (not accessible by address)
17 777 656—17 777 640	User Instruction PAR, Reg. 0-7
17 777 616—17 777 600	User Instruction PDR, Reg. 0-7
17 777 576	MM Status Register 2 (SR2)
17 777 574	MM Status Register 1 (SR1)
17 777 572	MM Status Register 0 (SR0)
17 777 570	Display Register
17 777 566—17 777 560	Console Terminal SLU
17 776 500—17 776 506	SLU 2
17 772 516	MM Status Register 3 (SR3)
17 772 356—17 772 340	Kernel Instruction PAR, Reg. 0-7
17 772 316—17 772 300	Kernel Instruction PDR, Reg. 0-7
17 770 372—17 770 200	UNIBUS Map Registers (optional with UNIBUS map)

▪ Memory System

Memory Management

The memory management hardware is standard with the PDP-11/24 computer. This hardware relocation and protection facility can convert the 16-bit program virtual addresses to 22-bit physical addresses. The unit may be enabled or disabled under program control. There is a small speed advantage when the unit is in the 16-bit mode.The basic function of relocating memory and protecting individual programs from one another is described in detail in the *PDP-11 Architecture Handbook*.

MOS Memory with Error Correcting Code and Optional Battery Backup

MOS memory with error correcting code (ECC) is identical to the memory available for the PDP-11/44 and PDP-11/84 systems. For an explanation of ECC and optional battery backup, refer to the descriptions in the PDP-11/84 chapter.

UNIBUS Map

The UNIBUS map responds as memory on the UNIBUS. It is the hardware relocation facility for converting the 18-bit UNIBUS addresses to 22-bit addresses. The relocation mapping may be enabled or disabled under program control. The UNIBUS map is an optional feature of the PDP-11/24, except when used in conjunction with the one-Mbyte MS11-P memory. It then becomes a standard feature.

▪ PDP-11/24 Backplane Configuration

The PDP-11/24 backplane (see Figure 4-6) consists of nine slots. Slot 1 is reserved for the M7133 CPU module. Slot 2 can contain memory or the UNIBUS map module. Additional memory can be configured in slots 3-6. (In the 5.25-inch box, the total number of MS11-L memory modules cannot exceed three; only one MS11-P memory module can be configured in the 5.25-inch box). The UNIBUS map is generally required for configurations with more than 256 Kbytes of memory. Slot 9 contains either the M9312 bootstrap/terminator, the M9302 terminator (if the UNIBUS map option is installed), or the UNIBUS cable.

The PDP-11/24 uses MS11-LB (128 Kbytes), MS11-LD (256 Kbytes) or MS11-PB (1 Mbyte) MOS ECC memory. These have the characteristics outlined in Table 4-2.

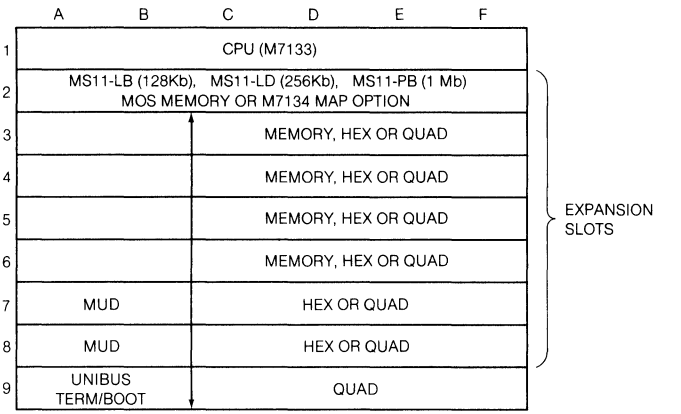


Figure 4-6 ▪ PDP-11/24 Backplane Configuration

Table 4-2 ▪ Memory Modules

	Size (Bytes)	Access Time (nsec)	Cycle Time (nsec)	Refresh
MS11-LB (18-bit)	128 Kbytes	360 for DATI 95 for DATO	450	560 ns every 12.5 μs
MS11-LD (18-bit)	258 Kbytes	360 for DATI 95 for DATO	450	560 ns every 12.5 μs
MS11-PB (22-bit)	1 Mbyte	490 for DATI (720 w/err) 100 for DATO	680 580	675 ns every 13.3 μs

▪ Console

The console terminal of the PDP-11/24 provides two functions.

- Communication with the operating software
- Debugging of hardware and software

When the console terminal is communicating with the operating software, it is in program mode. When the PDP-11/24 is stopped, the console terminal is in console-ODT mode.

The two modes are summarized in Table 4-3. Various conditions can switch the console terminal between modes. These conditions are diagrammed in Figure 4-7.

Table 4-3 ▪ PDP-11/24 Console Terminal Modes

	Program Mode	Console Mode
Microprogram running?	Yes	Yes
Macroprogram running?	Yes	No

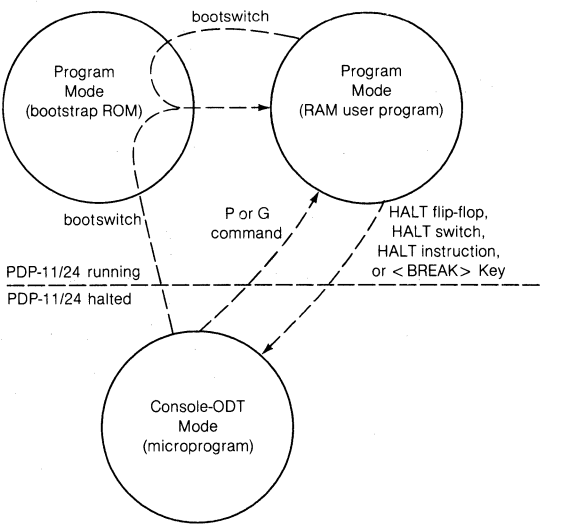


Figure 4-7 ▪ Conditions That Switch the Console Terminal between Modes

Program Mode Functions

While the console terminal is operating in program mode, it is under the sole control of the operating system software, and most of its functions are defined by that software.

▪ LEAVING PROGRAM MODE

The user can leave program mode and enter console-ODT in one of four ways—by pressing the <BREAK> key while the PDP-11/24 keyswitch is in the <LOCAL> position, by placing the HALT/RUN/BOOT switch in the HALT position (again while the keyswitch is in the <LOCAL> position), by the PDP-11 program executing a HALT instruction while in kernel mode, or by the HALT flip-flop being set (from a previous H command.)

Console ODT

Besides machine setup, the console mode can also be used to examine or modify the contents of memory or UNIBUS device registers. Programs may also be stopped, started, or single-stepped. These functions are implemented by a microprogram contained within the F-11 processor chip. During the execution of these functions, the PDP-11/24 CPU is stopped. That is, although the F-11 microprogram is still running, no PDP-11 macroinstructions are being executed.

These functions are collectively referred to as console ODT (octal debugging technique) and are generally compatible with both ODT-11 and the console ODT functions provided by the LSI-11, the MicroPDP-11/23, and the PDP-11/84.

Console ODT can examine any read-only or read/write address in the first 248 Kbytes of main memory or 8 Kbytes of I/O page. Console ODT can also modify any read/write (not write-only) location. Locations that read all zeros (such as the console XBUF) can also be modified. The ODT commands are summarized in Table 4-4.

Table 4-4 • Console ODT Commands

Command	Symbol	Function
Slash	n/	Opens the specified location (n) and outputs its contents. n is an octal number.
Carriage Return	<CR>	Closes an open location.
Line Feed	<LF>	Closes an open location and then opens the next contiguous location.
Internal Register Designator	$n, Rn, or rn	Opens a specific processor register (n). n is an integer from 0 to 7 or the character S.
Processor Status Word Designator	S or s	Opens the PSW—must follow $ or R command.
Go	G or g	Starts program execution.
Proceed	P or p	Resumes execution of a program.
Binary Dump	Control-S	Manufacturing use only.
Toggle Halt flip-flop	H	Sets or clears an internal flip-flop that acts in parallel with the HALT position of the HALT/CONT/BOOT switch.

A detailed description of the console ODT commands for the PDP-11/24 is found in Appendix B.

▪ LEAVING CONSOLE ODT MODE

You can leave console-ODT mode and return to program mode via

- The P command
- The G command
- The boot switch

The P or G command will exit you back to program mode. (The G command is used for a total restart, while the P command allows you to proceed). The bootswitch will reboot the I/O device selected by an internal switchpack.

▪ Address Specification

All addresses must be entered by users with all 18 bits specified, regardless of whether the MMU is present or not. For example, if a user desires to open the RCSR of Serial Line Unit 1, the user must enter 777 560, not 177 560. Eighteen-bit addresses must also be used to access memory greater than 32 Kwords. Leading zeros need not be typed. Console ODT cannot access memory above 248 Kbytes (124 Kwords).

Processor I/O Addresses

Certain processor and MMU registers have I/O addresses assigned to them for programming purposes. If referenced in console ODT, the PSW responds to its bus address, 777 776. Processor registers R0 through R7 do not respond to bus addresses 777 700 through 777 707 if referenced in console ODT (i.e., timeout occurs).

The MMU contains status registers and PAR/PDR pairs. Any of these registers can be accessed from console ODT by entering its bus address.

Example: @**777572**/000001 <SPACE> (User's entry in **boldface** type)

In this case, memory management status register 0 is opened and the memory management enable is seen to be set. The internal display register (777 570) can not be accessed with ODT because the register is write-only and ODT always first attempts to read the register.

Stack Pointer Selection

Accessing kernel and user stack pointer registers is accomplished in the following way. Whenever R6 is referenced in ODT, it accesses the stack pointer specified by the PSW current mode bits (PSW<15-14>). This is done for convenience. If a program operating in kernel mode (PSW<15-14>=00) is halted and R6 is opened, the kernel stack pointer is accessed.

Similarly, if a program is operating in user mode, R6 accesses the user stack pointer. If a specific stack pointer is desired, PSW <15-14> must be set by the

user to the appropriate value and then the R6 command can be used. If an operating program has been halted, the original value of PSW <15-14> must be restored in order to continue execution.

Example: PSW = 140000

@R6/123456<SPACE>

The user mode stack pointer has been opened.

@RS/140000<SPACE>**0**<**CR**><CR><LF> (switch to kernel mode)

@R6/123456<SPACE><**CR**><CR><LF> (read the kernel stack pointer)

@RS/000000<SPACE>**140000**<**CR**><CR><LF> (return to user mode)

@P

In this case, the kernel mode stack pointer was desired. The PSW was opened and PSW<15-14> was set to 00 (kernel mode). Then R6 (kernel stack pointer) was examined and closed. The original value of PSW<15-14> was restored and then the program was continued using the P command.

If PSW<15-14> is set to 01, another unique register exists in the processor, but is reserved for future Digital use.

The floating-point accumulators, which are also in the MMU chip, cannot be accessed from console ODT. Only floating-point instructions can access these registers.

Entering Octal Digits

When the user is specifying an address of data, console ODT will use the last six octal digits if more than six have been entered. The user need not enter leading zeros for either address or data; console ODT forces zeros as the default. If an odd address is entered, the low-order bit is ignored and a full 16-bit word is displayed.

ODT Timeout and Parity Errors

If the user specifies a nonexistent address, console ODT responds to the error by printing ?<CR><LF>@. If a parity error is indicated while opening a location, console ODT also prints ?<CR><LF>@.

Invalid Characters

Console ODT will recognize, with the exception of H, uppercase and lowercase characters as commands. Any character that console ODT does not recognize during a particular sequence is echoed (with the exception of ASCII 0, 2, 10, or 12) and console ODT prints a ?<CR><LF>@.

▪ Terminal Serial Line

The PDP-11/24 contains two serial line interfaces as a standard feature. The first interface (the console interface) is used to control the PDP-11/24 hardware and the operating system software.

While controlling the PDP-11/24 hardware, the console interface is said to be in console mode and is operated by a fixed program. While controlling, and in turn being controlled by, operating system software, the console interface is said to be in program mode. The selection of console or program mode is made via the PDP-11/24 front panel and by special characters typed at the console terminal. Use of the console mode and program mode is described earlier in this chapter.

Serial Line Unit Timing Considerations

The serial line unit timing considerations for the PDP-11/24 are identical to the PDP-11/44. Therefore an explanation of the UART (Universal Asynchronous Receiver/Transmitter) subsystem may be found in the PDP-11/44 chapter.

Terminal Serial Line Unit Control Registers (SLU 1)

There are four Terminal SLU registers that follow. All unused or write-only bits are zero when examined

▪ RECEIVER STATUS REGISTER (TERM RCSR)
Bits: 15–8
Function: Unused

Bit: 7 (Read Only)
Name: RECEIVER DONE
Function: Set when an entire character has been received and is ready for transfer to the UNIBUS. Cleared by addressing (Read or Write) RBUF or by INIT. Starts an interrupt sequence when RECEIVER INTERRUPT ENABLE (bit 6) is also set.

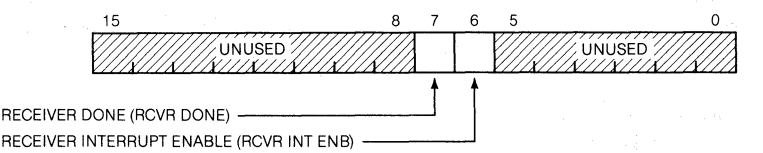


Figure 4-8 ▪ 17 777 560 Receiver Status Register

Bit: 6 (Read/Write)
Name: RECEIVER INTERRUPT ENABLE
Function: Cleared by INIT. Starts a priority 4 interrupt sequence when RECEIVER DONE is set.

Bits: 5–0
Function: Unused

- RECEIVER DATA BUFFER (TERM RBUF)

Bit: 15 (Read Only)
Name: ERROR
Function: Logical OR of OVERRUN ERROR, FRAMING ERROR, and PARITY ERROR. Cleared by removing the error conditions. ERROR is not tied to the interrupt logic, but RECEIVER DONE is.

Bit: 14 (Read Only)
Name: OVERRUN
Function: Set if the previously received character is not read (RECEIVER DONE not reset) before the present character is received. This indicates that the previous character(s) have been lost.

Bit: 13 (Read Only)
Name: FRAMING ERROR
Function: Set if the character read has no valid stop bit. Also used to detect break.

Bit: 12 (Read Only)
Name: RECEIVE PARITY ERROR
Function: Set if received parity does not agree with the expected parity. Always zero if no parity is selected.

NOTE

Error conditions remain present until the next character is received, at which time the error bits are updated. INIT does not necessarily clear the error bits.

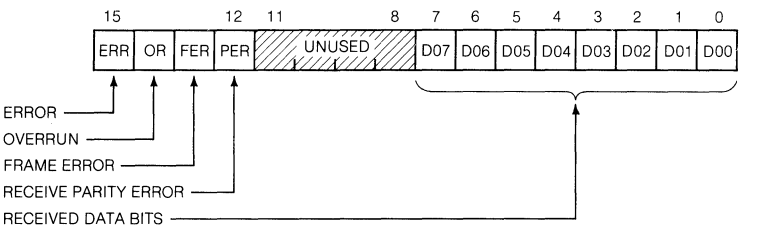


Figure 4-9 ▪ 17 777 562 Receiver Data Buffer

Bits: 11–8
Function: Unused

Bits: 7–0 (Read Only)
Name: RECEIVED DATA
Function: These bits contain the character just read. If fewer than eight bits are selected, the buffer will be right-justified with the unused bits read as zeros. Not cleared by INIT.

- TRANSMITTER STATUS REGISTER (TERM XCSR)

Bits: 15–8
Function: Unused

Bit: 7 (Read Only)
Name: TRANSMITTER READY
Function: Set by INIT. Cleared when XBUF is loaded; set when XBUF can accept another character. When set, it will start an interrupt sequence if TRANSMITTER INTERRUPT ENABLE is also set.

Bit: 6 (Read/Write)
Name: TRANSMITTER INTERRUPT ENABLE
Function: Cleared by INIT. As per Receiver Interrupt Enable, when set it will start a priority 4 interrupt sequence if TRANSMITTER READY is also set.

Bits: 5–3
Function: Unused

Bit: 2 (Read/Write)
Name: MAINTENANCE
Function: Cleared by INIT. When set, connects the serial output of the TRANSMITTER into the serial input of the RECEIVER, in place of the normal serial input from the terminal. It also forces the receiver to run at the same speed as the transmitter.

Bits: 1–0
Function: Unused

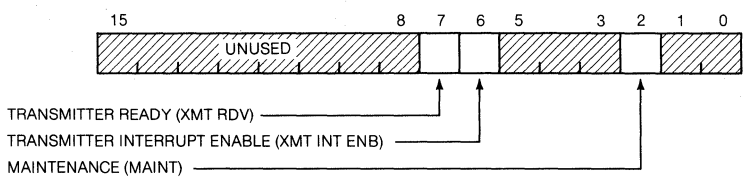


Figure 4-10 ▪ 17 777 564 Transmitter Status Register

- TRANSMITTER DATA BUFFER (TERM XBUF)

Bits: 15–8
Function: Unused

Bits: 7–0 (Write Only)
Name: TRANSMITTER DATA BUFFER
Function: If fewer than eight bits are selected, then the character must be right-justified. The character to be transmitted is written into this register.

Second Serial Line Unit Registers (SLU 2)

The second serial line unit is a general-purpose serial line interface. It may be used for a variety of purposes:

- Connection of a serial line-printer
- Connection of a TU58 cartridge tape drive
- Connection of a modem

This interface is not recommended for use with a high-speed (> 1200 baud) interactive terminal, such as a VT220 or VT240/z). The four SLU2 registers follow.

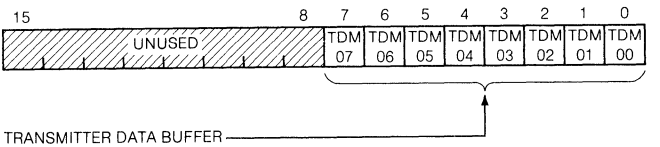


Figure 4-11 • 17 777 566 Transmitter Data Buffer

- RECEIVER CONTROL STATUS REGISTER (RCSR)

Bits: 15–8
Function: Unused

Bit: 7 (Read Only)
Name: RECEIVER DONE
Function: Set when a complete character is contained in the RBUF. Cleared when the RBUF is addressed or when an INITIALIZE operation occurs. Initiates the interrupt sequence when the RECEIVER INTERRUPT ENABLE (bit 6) is set.

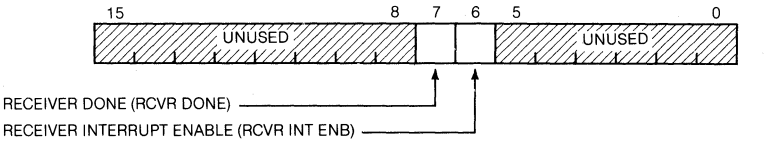


Figure 4-12 ▪ 17 776 500 Receiver Control/Status Register

Bit: 6 (Read/Write)
Name: RECEIVER INTERRUPT ENABLE
Function: Set by program to allow a priority 4 interrupt sequence to be initiated by the RECEIVER DONE (bit 7).

Bits: 5–0
Function: Unused

- RECEIVER BUFFER REGISTER (RBUF)

Bit: 15 (Read Only)
Name: ERROR
Function: Set when the OR ERROR (bit 14), FR ERROR (bit 13) or the PAR ERROR (bit 12) is set. Cleared by the reception of new and correct data.

Bit: 14 (Read Only)
Name: OVERRUN ERROR
Function: Set if the character in the RBUF has not been read before another character is received. Cleared by an INITIALIZE operation or when the RBUF is emptied.

Bit: 13 (Read Only)
Name: FRAMING ERROR
Function: Set when the character read in RBUF does not have a valid stop bit(s). Cleared when a valid character is received. This bit may indicate an error in transmission or the reception of a BREAK character.

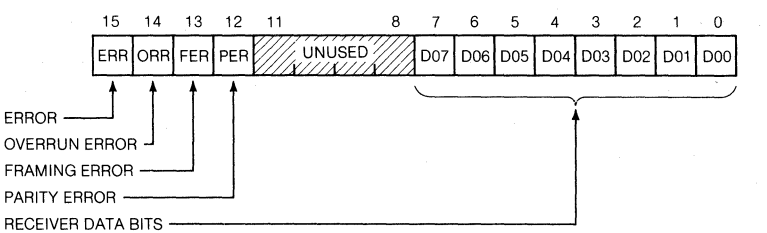


Figure 4-13 ▪ 17 776 502 Receiver Buffer Register

Bit: 12 (Read Only)
Name: PARITY ERROR
Function: Set when the parity of the character read in the RBUF is incorrect relative to the parity mode selected. Cleared when the parity of the next character is validated.

Bits: 11–8
Function: Unused

Bits: 7–0 (Read Only)
Name: RECEIVED DATA
Function: These bits constitute the data characters received from the SLU 2.

- TRANSMITTER CONTROL/STATUS REGISTER (XCSR)

Bits: 15–8
Function: Unused

Bit: 7 (Read Only)
Name: TRANSMITTER READY
Function: Set when the XBUF is ready to accept a character or when an initialize operation occurs. Setting the bit indicates an interrupt sequence if the TRANSMITTER INTERRUPT ENABLE (bit 6) is set. Cleared when a character is written into the XBUF.

Bit: 6 (Read/Write)
Name: TRANSMITTER INTERRUPT ENABLE
Function: Set by program. Enables a priority 4 interrupt sequence to be initiated if the TRANSMITTER READY (bit 7) is set. Cleared by the program.

Bits: 5–1
Function: Unused

Bit: 0 (Read/Write)
Name: BREAK
Function: Set by the program. When set, a continuous space is transmitted equivalent to sending a null character with no stop bits (framing error). May be diabled by removing a jumper on the M7096 module.

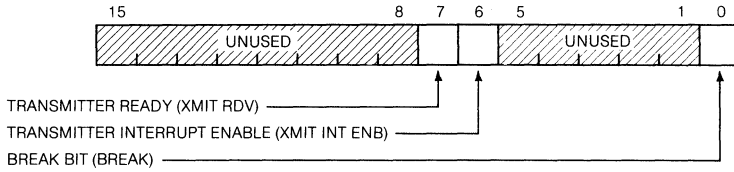


Figure 4-14 ▪ 17 776 504 Transmitter Control/Status Register

▪ TRANSMITTER DATA BUFFER REGISTER (XBUF)

Bits: 15–8
Function: Unused

Bits: 7:0 (Write Only)
Name: TRANSMITTER DATA
Function: These bits constitute the data characters to be transferred to the SLU 2.

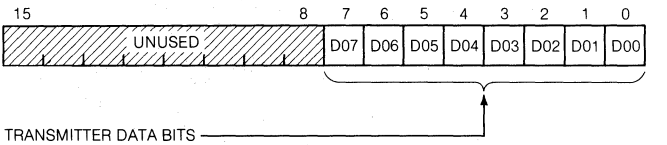


Figure 4-15 ▪ 17 776 506 Transmitter Data Buffer Register

▪ Line Clock

The PDP-11/24 includes a line-time clock as standard equipment. This clock provides interrupts synchronized with the cycles of the ac power line. By counting these interrupts, this allows the operating system software to keep realtime.

Clock Status Register (LKS)

Bits: 15–8
Function: Unused

Bit: 7 (Read/Clear)
Name: LINE CLOCK MONITOR
Function: Set only by the line frequency clock signal and cleared only by the program or the line clock interrupt sequence. Set by INIT.

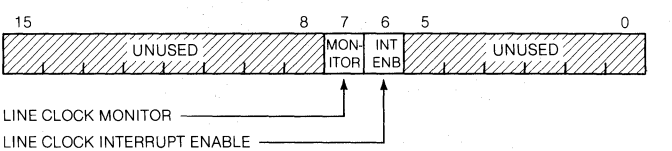


Figure 4-16 ▪ 17 777 546 Clock Status Register

Bit: 6 (Read/Write)
Name: LINE CLOCK INTERRUPT ENABLE
Function: Cleared by INIT. When set, starts an interrupt sequence if LINE CLOCK MONITOR is also set.

Bits: 5–0
Function: Unused

▪ Address and Vector Assignments

Integral to the PDP-11/24 CPU are the two serial line units and the clock. The serial line units follow the same address and vector assignments as the KL11, DL11-A,-B,-C,-D,-W, and the KW11-L, respectively. SLU 1 is for the system console and has fixed addresses and vectors. SLU 2, which may be used for asynchronous devices, has switch-selectable contiguous addresses and vectors. The realtime clock has a fixed address and vector.

Table 4-5 ▪ Address and Vector Assignments

	Address	Vector	Priority
Console (SLU #1)	17 777 560 17 777 562 17 777 564 17 777 566	60/64	BR4
(SLU #2)	17 776 500 17 776 502 17 776 504 17 776 506	300/304	BR4
Line Clock	17 777 546	100	BR6

▪ Specifications

Packaging

A basic PDP-11/24 consists of either a 5.25-inch or a 10.5-inch box with a 9-slot backplane, power supply, CPU, 128-Kbyte, 256-Kbyte, or 1-Mbyte memory.

There are prewired areas within the backplane for expansion with optional equipment.

Component Parts

The basic PDP-11/24 system contains

- Standard Equipment
  - —PDP-11/24 CPU.
  - —Memory management.
  - —Bootstrap loader.
  - —Line-frequency clock.
  - —Second serial line interface.
  - —Terminal interface.
  - —128-Kbyte, 256-Kbyte, or 1-Mbyte ECC MOS memory.
  - —BA11-L or BA11-A box with power supply.

- Prewired Expansion Space for Optional Equipment.
  - —Floating-point unit.
  - —SPC slots for peripherals, up to 6 hex height and 1 quad height (depending on the memory configuration).
  - —768-Kbyte parity MOS memory (up to 1024 Kbytes maximum).
  - —4 system units of open space in BA11-A Box.

Other Specifications

- AC POWER

5.25-in Box	104-127 Vrms, 47-63 Hz, 1 phase power, 5 amps rms maximum @ 120 Vac
10.5-in Box	90-128 Vrms, 47-63 Hz, 1 phase power, 9 amps rms maximum @ 240 Vac

- SIZE

5.25-in Box	Cabinet is 13.5 cm high by 48 cm wide by 69 cm deep (5.25 in by 19 in by 25 in)
10.5-in Box	Cabinet is 26.3 cm high by 42.4 cm wide by 66 cm deep (10.35 in by 16.62 in by 26 in)

- WEIGHT

5.25-in Box	20 Kg (45 lbs)
10.5-in Box	32 Kg (70 lbs)

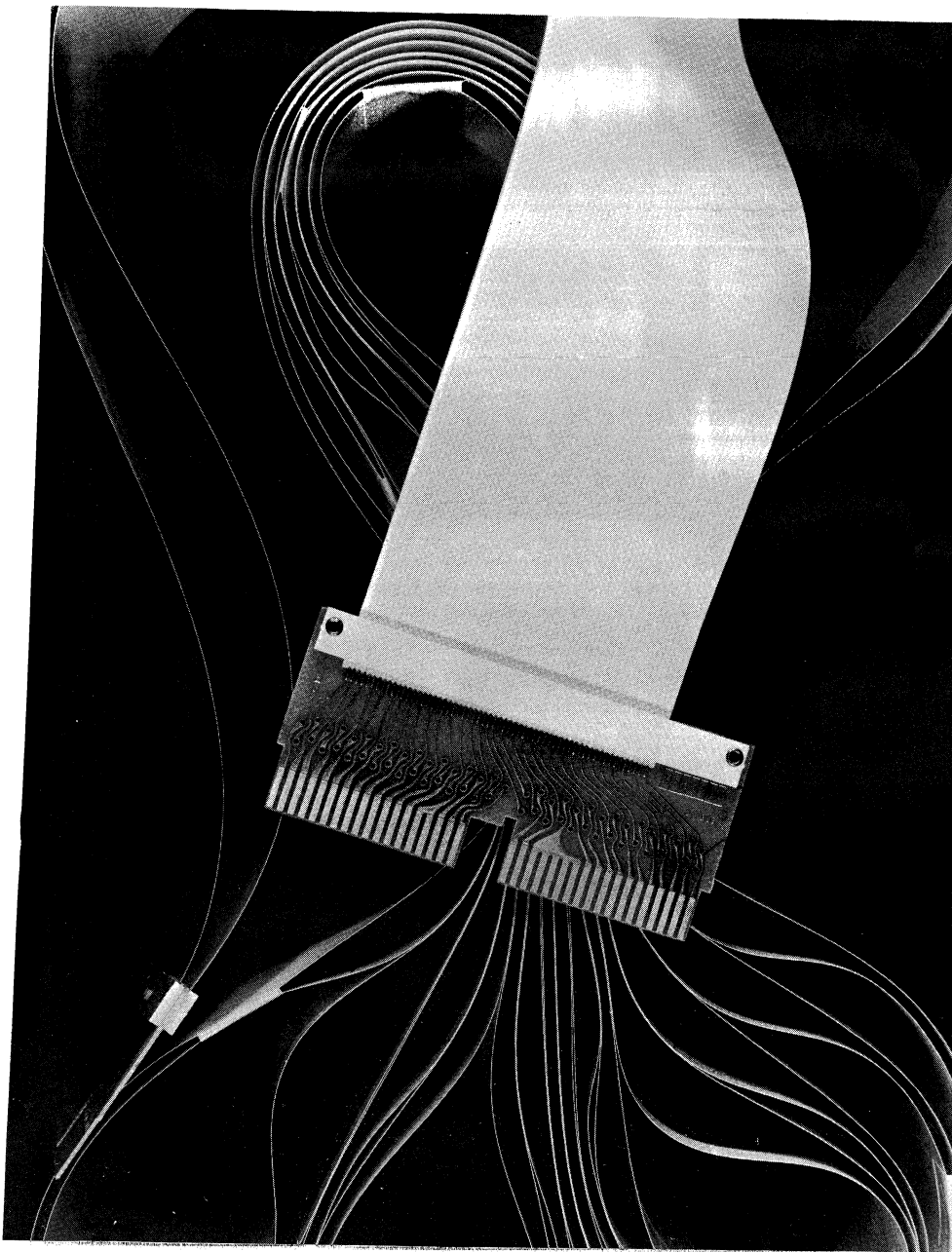
- OPERATING ENVIRONMENT

The 5.25-inch and 10.5-inch CPU boxes have the same operating and nonoperating environment specifications.

Temperature:	5°C to 50°C (41°F to 122°F)
Humidity:	10% to 95% with maximum wet bulb of 32°C (89.6°F) and minimum dew point of 2°C (36°F)
Altitude:	To 2.4 Km (8,000 ft) noncondensing

- NONOPERATING ENVIRONMENT

Temperature:	–40°C to 80°C (–40°F to 176°F)
Humidity:	To 95% noncondensing
Altitude:	To 9.1 Km (30,000 ft)



▪ Characteristics of the UNIBUS

All computers must contain some method of interfacing with the outside world. For some computers, this requires a collection of integrated input/output (I/O) controllers with special connections to the processor and/or memory. For other computers, this requires a collection of wires, and a protocol explaining how to use these wires to communicate with the rest of the computer. The set of wires is called a *bus*. The UNIBUS is the particular bus used by many of the PDP-11 and VAX-11 computers from Digital.

Some of the major characteristics of the UNIBUS are listed below, and discussed in subsequent paragraphs.

- Nonmultiplexed bus
- Strict master/slave relationship
- Partially distributed arbitration
- Overlapped arbitration and data transfer
- Asynchronous operation
- 18 address bits
- Word or byte operations
- Parity error information from slaves

Nonmultiplexed Bus

Most buses contain the concepts of address and data. All things on the bus are uniquely accessed by means of an *address*, or a series of addresses. The address on the bus is very much like a telephone number or an address on an envelope. Once a unique device has been addressed (selected), *data* may be exchanged with that device. This data would correspond to the phone conversation, or the letter itself.

Some buses first use the same physical set of wires to carry the address, followed by one or more items of data. These buses are called *multiplexed buses* (see Figure 5-1). The Digital Q-bus is an example of a multiplexed bus. Other buses use separate sets of wires to carry the address and data. These buses are called *nonmultiplexed buses* (see Figure 5-2). The UNIBUS is an example of a nonmultiplexed bus.

Because an address and data can be carried simultaneously, a nonmultiplexed bus is usually faster, particularly if each new address bears no relationship to any preceding address. However, multiplexed buses share the same set of wires for address and data so they tend to be physically smaller and thereby less expensive. If addresses are usually presented in a particular order (e.g., ascending), the next address can be predicted rather than explicitly transmitted, and the performance of the multiplexed bus can approach the performance of the nonmultiplexed bus.

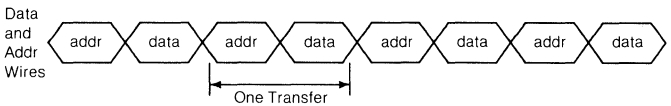


Figure 5-1 ▪ A Multiplexed Bus (like the Q-bus)

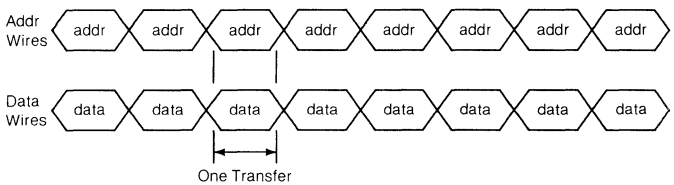


Figure 5-2 ▪ A Nonmultiplexed Bus (like the UNIBUS)

Strict Master/Slave Relationship

Most transfers on the UNIBUS consist of a single device (called the *bus master*, or just *master*) requesting another device (the *slave*) to perform some operation. Other buses may use the terms *commander* and *responder* as synonyms for master and slave. No direction of data flow is implied here—the master may be requesting that the slave read or write data.

Partially Distributed Arbitration

The UNIBUS, like most buses, allows only one device at a time to be the bus master. This implies that there must be some method of selecting a bus master from among all of the devices requesting the bus. The process that performs this selection is called *priority arbitration*, and the bus is usually granted to the requester with the highest priority (according to some pre-arranged scheme).

Arbitration on the UNIBUS is performed using a two-dimensional scheme. A typical example is shown in Figure 5-3. The first dimension is the specific priority level of the request. The UNIBUS may be requested at any of five different priority levels. The highest priority level is named *nonprocessor request* (NPR), so named because devices requesting use of the UNIBUS at this priority level do not need any assistance from the central processor. This is somewhat analagous to *direct memory access* (DMA), although the UNIBUS allows NPR transfers between any two devices, not just between a device and memory. The lower four levels are simply called *bus request 7* (BR7) through *bus request 4* (BR4). Devices request use of the UNIBUS via BR levels in order to interrupt a processor (described later).

All requests are passed to a central *bus arbitrator*. This central bus arbitrator is usually included in the PDP-11 processor or VAX-11 UNIBUS adapter or interface. In any case, it is always located at the *front end* of the UNIBUS. The

arbitrator determines the highest requested level from among the five available and issues a *bus grant* for that level, if appropriate. The grant then travels towards the *back end* of the UNIBUS. *Nonprocessor grant* (NPG) is given in response to NPR, bus grant 7 (BG7) in response to BR7, and so forth.

The second dimension of the arbitration is performed when more than one device requests the UNIBUS at the same priority level. Now the grant is passed from device to device, towards the back end of the UNIBUS (this is referred to as *daisychaining*). Each device, in turn, considers whether it wants to use the UNIBUS. A device that does not want to use the UNIBUS passes the grant on to the next device along the UNIBUS. A device that does want to use the UNIBUS does not pass the grant (i.e., it *blocks the grant*) and the device wins the arbitration. Thus arbitration among more than one device on a particular level is distributed among the devices.

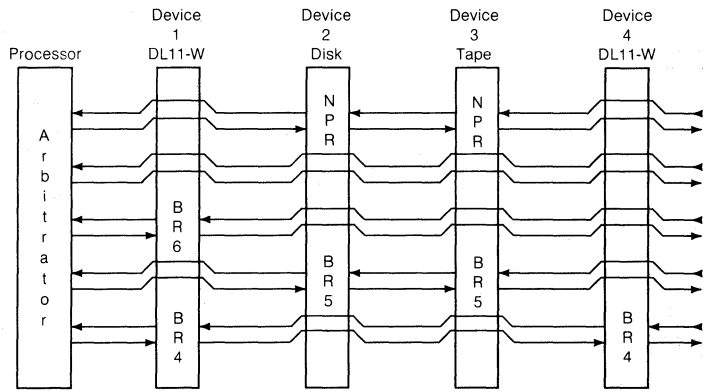


Figure 5-3 ▪ UNIBUS Two-dimensional Priority

Figure 5-3 illustrates the beginning of a typical UNIBUS. Here, the first DL11-W (Device 1) can request the UNIBUS at priority levels BR6 or BR4; both the disk controller (Device 2) and the tape controller (Device 3) can request at NPR or BR5; the second DL11-W (Device 4) can request only at BR4. No device in this illustration uses BR7.

The UNIBUS arbitrator considers the five levels of priority and grants the highest-level request. So in our example, the disk or the tape requesting at the NPR level always have priority over either of the DL11-Ws requesting at either BR6 or BR4.

On any given level (e.g., NPR), the grant is passed from device to device along the UNIBUS. At the NPR level, the disk has priority over the tape because the disk gets an opportunity to examine (and possibly block) the grant before passing it on to the tape. Similarly, at the BR4 level, the first DL11-W has priority over the second DL11-W.

Overlapped Arbitration and Data Transfer

As mentioned previously, before a prospective bus master may use the bus, it must win the arbitration (ensuring that it is the highest priority requester of the bus). Some buses can do only one thing at a time (e.g., arbitrate or transfer data), but not both. This is illustrated in Figure 5-4. The UNIBUS, on the other hand, allows the next master to be selected (arbitrated) while the current master is still transferring data. This is referred to as *overlapped arbitration* and greatly contributes to the performance of the UNIBUS. See Figure 5-5.

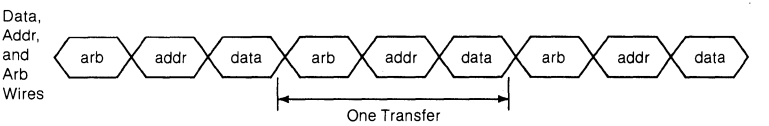


Figure 5-4 ▪ A Non-overlapped Arbitration Bus

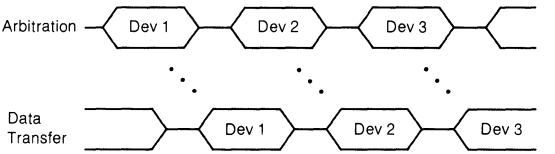


Figure 5-5 ▪ An Overlapped Arbitration Bus (like the UNIBUS and Q-bus)

Asynchronous Operation

A bus may be synchronous or asynchronous. Synchronous buses partition time into parcels of fixed duration, usually called cycles. These cycles are defined by one or more clocks on the bus. Every operation on the bus must take place in an integral number of these cycles. Some buses require that every operation must take place within a single bus cycle. A synchronous bus of this sort is illustrated in Figure 5-6. Here, every rising edge of the system clock implies that a valid address is on the bus; every falling edge of the system clock implies that the data has replaced the address. Although synchronous buses may be very simple

or very fast (although usually not both), this fixed timing imposes severe constraints on the system designer. The duration of the bus cycle must be long enough to allow for the slowest peripheral, yet short enough to get reasonable system performance.

Other synchronous systems allow a device to take multiple bus cycles for one transfer. A bus of this sort is illustrated in Figure 5-7. Here, a *wait* (or *stall*) line freezes the system, allowing slow slaves more time to process the data. This eases the timing constraints somewhat but does so at the expense of a more complex design. Also, when a *wait* or *stall* must occur, it must occur in whole-cycle "chunks," even if the data was late by only a few nanoseconds. Stalling by whole cycles degrades the performance of this synchronous bus.

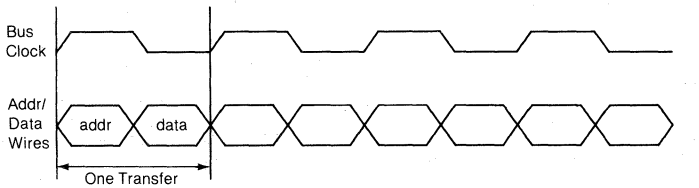


Figure 5-6 ▪ A Single-cycle Synchronous Bus

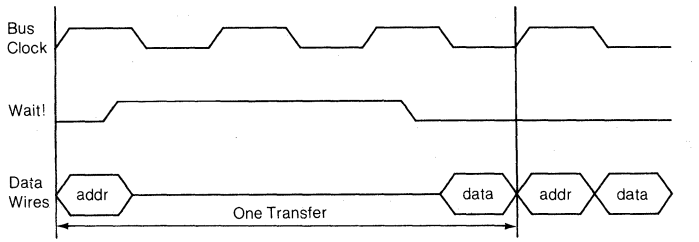


Figure 5-7 ▪ A Multicycle Synchronous Bus

The UNIBUS, on the other hand, is completely asynchronous. No master clock times the operations of the UNIBUS. Rather, the selected master and slave devices exchange *handshaking* signals to indicate the presence of information on the bus. This handshaking allows each bus cycle to run as quickly as the particular master/slave pair will allow. Figure 5-8 shows a series of read data cycles on the UNIBUS (i.e., the master has asked the slave to provide data to the master). Note that the master first asserts the address of the desired slave, then *address strobe* (actually called MASTER SYNC or BUS MSYN L). The slave then

looks up the desired data and places it on the data lines along with a signal that combines the functions of *data strobe* and *address acknowledge* (actually called SLAVE SYNC or BUS SSYN L). No fixed timing is required between BUS MSYN L and BUS SSYN L; fast slaves can respond quickly while slow slaves can take more time to produce the requested data.

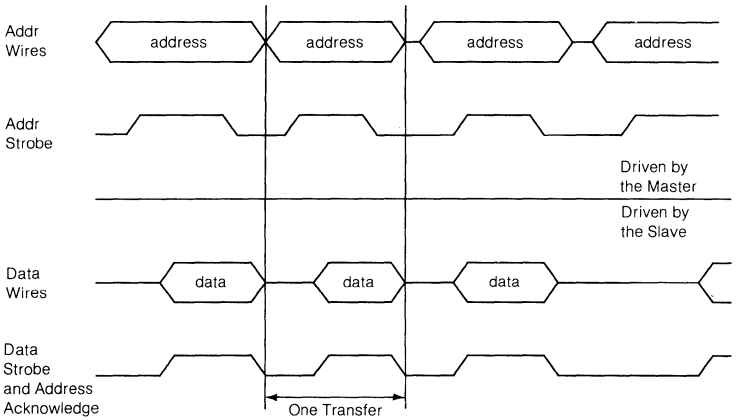


Figure 5-8 ▪ An Asynchronous Bus (like the UNIBUS)

18 Address Bits

The total number of unique data that can be addressed on any bus is dependent on the number of bits used to form the address. The UNIBUS uses 18 bits to specify an address. Each address represents a *byte* of data. Thus the UNIBUS allows 2^{18} = 256K* Unique Byte Addresses

Of these 256K byte addresses, 248K are reserved for system main memory, or access to system main memory, and 8K are used for access to I/O devices.

Word or Byte Operations

The UNIBUS allows data to be accessed as a 16-bit word or an 8-bit byte.

Parity Error Information from Slaves

If the slave contains logic to detect parity errors (e.g., a MOS memory), the slave can pass a parity error indication to the master via a predefined UNIBUS signal. The UNIBUS itself does not include parity checking of any bus signals.

*K = 1024

▪ UNIBUS Block Diagram

The UNIBUS may be viewed as three separate but cooperating buses. Please refer to Figure 5-9.

- Initialization and shutdown bus
- Arbitration bus
- Data transfer bus

The first bus contains signals used to initialize the system or to signal that power has failed and the system must now be shut down in an orderly fashion. Any processor may drive the system initialization signal (BUS INIT L), while power supplies are generally responsible for driving the system shutdown signals (BUS AC LO L and BUS DC LO L). Every device monitors one or more of these signals.

The third bus allows the exchange of data between a selected master device and a selected slave device.

The second bus controls who may have access to the data transfer bus. A central device called the *bus arbitrator* receives *requests* for use of the UNIBUS, and replies with *grants* (permission to use the UNIBUS).

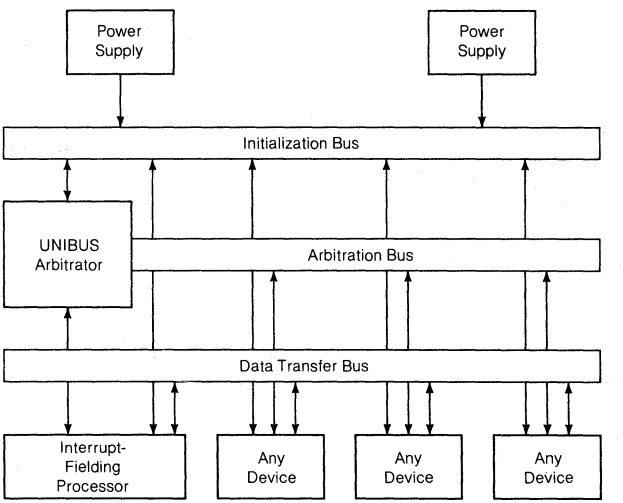


Figure 5-9 ▪ UNIBUS Block Diagram

The UNIBUS may contain more than one processor but only one processor is designated to be in charge of handling (servicing) interrupts. This processor is called the *interrupt-fielding processor* (IFP). The arbitrator may, on its own authority, issue an NPG in response to an NPR, since a nonprocessor request by definition does not allow a processor to be interrupted. If, however, the request is a BR (which would interrupt the IFP), the arbitrator must first consult the IFP for permision to issue the BG, since the IFP may not allow the interrupt.

▪ UNIBUS Data and Address Organization

Addresses

Addresses on the UNIBUS are 18-bits wide and represent the address of a particular byte. The bits are numbered as shown in Figure 5-10, address bit 00 being the Least Significant Bit (LSB) and address bit 17 being the Most Significant Bit (MSB).

Words are addressed on even-byte boundaries. That is, address bit 00 is always equal to zero (=0) when accessing a word.

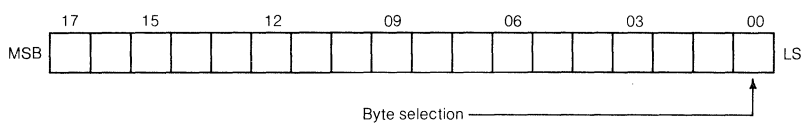


Figure 5-10 ▪ UNIBUS Address Format

Data

Data words on the UNIBUS are 16-bits wide. The bits are numbered as illustrated in Figure 5-11, bit 00 being the LSB and bit 15 being the MSB. During word operations, address bit 00 must equal zero.

Each data word may also be construed as two data bytes. See Figure 5-12. Data on the UNIBUS is never *justified*. That is, data bits 07-00 always form the even bytes (i.e., bytes with address bit 00=0) and data bits 15-08 always form the odd bytes (i.e., bytes with address bit 00=1). When two bytes are construed as a word, the even byte forms the low-order bits of the word and the odd byte forms the high-order bits of the word.

The UNIBUS does not define a byte-read operation. Reads always produce a word (two bytes) of data, and it is the responsibility of the master to select the correct byte, based on address bit 00. To read from an even byte, you must take your data from the lines representing word bits 07-00. To read from an odd byte, you must take your data from the lines representing word bits 15-08. For read operations, address bit 00 is never actually asserted on the bus.

The UNIBUS does define a byte-write operation. Here, it is the responsibility of the slave to update the selected byte of the data word (based on address bit 00) while preserving the unselected byte. The master must still drive data on the correct byte of the UNIBUS, and must now correctly assert address bit 00. In fact, the master's logic can be simplified by driving the same data onto both bytes of the UNIBUS simultaneously. The slave will select the correct byte and ignore the other byte.

Not all slave devices implement the byte-write operation for all addresses. Some addresses can be written only as entire words, and an attempt to write to a byte updates the entire word anyway, using whatever data happens to be on the other byte of the UNIBUS data lines. This is generally true of I/O device registers. Memories almost always support byte-write operations.

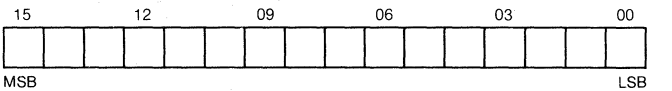


Figure 5-11 ▪ UNIBUS Data as a Word

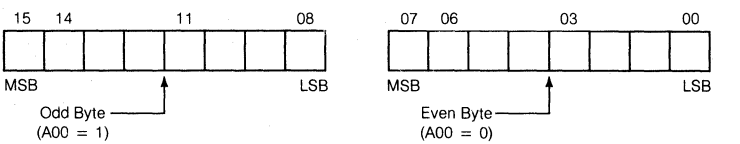


Figure 5-12 ▪ UNIBUS Data as Two Bytes

▪ Types of UNIBUS Data Transfers

The previous paragraphs introduced the three major subbuses of the UNIBUS, the form of a UNIBUS address, and the two forms of UNIBUS data. The following paragraphs will examine in more detail the functions of the Data Transfer Section.

Because the ultimate function of any bus is to move data from place to place, the Data Transfer Bus can be viewed as the central, most important part of the UNIBUS.

The Data Transfer Section implements five distinct types of operations.

- Read word
- Write word
- Write byte
- Read word (with write intent)
- Write vector

Read Word

The read word operation allows the current UNIBUS master to read one 16-bit word from a particular UNIBUS slave device. In UNIBUS terminology, this operation is called a DATI (Data-In). The direction ("In") is always taken with respect to the master device. For example, a processor (the master in this case) may wish to read a word of data contained in main memory (the slave). See Figure 5-13.

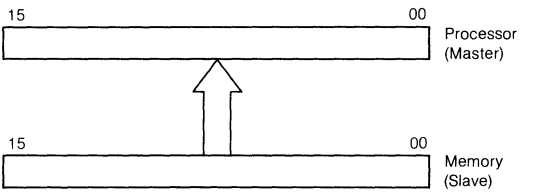


Figure 5-13 • DATI Used to Read a Word

This same operation is also used when the master wishes to read a byte. In this case, the slave still presents an entire word (two bytes) and the master simply selects the correct byte. See Figure 5-14.

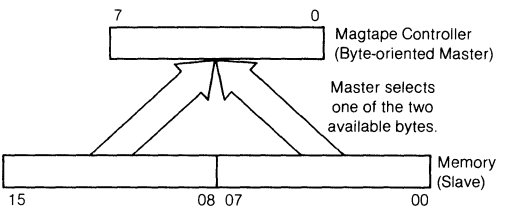


Figure 5-14 • DATI Used to Read a Byte

Write Word

This operation allows the current UNIBUS master to write one 16-bit word from the master to a particular UNIBUS slave device. In UNIBUS terminology, this operation is called a DATO (Data-Out). The direction ("Out") is once again taken with respect to the master device. In this example, a disk controller is writing into main memory. See Figure 5-15.

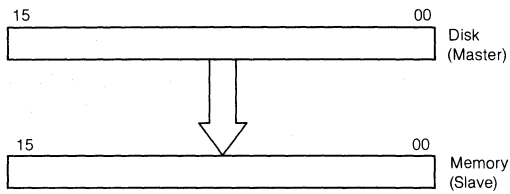


Figure 5-15 ▪ DATO (Write Word)

DATO always writes an entire 16-bit word. DATO can not write a single byte. The DATO must be directed at a word-address (i.e., an even byte address). In any case, most slave devices ignore address bit <00> during word operations; this forces the operation to an even address.

Write Byte

This operation allows the current UNIBUS master to write one 8-bit byte from the master to a particular UNIBUS slave device. In UNIBUS terminology, this operation is called a DATOB (Data-Out Byte). As usual, the direction ("Out") is taken with respect to the master device. See Figures 5-16 and 5-17.

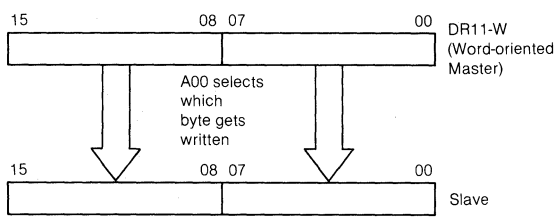


Figure 5-16 ▪ DATOB (Write Byte) for a Word-oriented Master

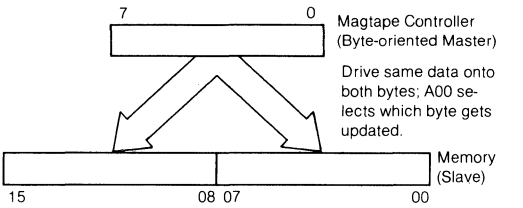


Figure 5-17 ▪ DATOB (Write Byte) for a Byte-oriented Master

DATOB attempts to write a single 8-bit byte. The other byte is to be left unmodified.

Read Word with Write Intent

Like DATI, this operation also allows the current UNIBUS master to read one 16-bit word from a particular UNIBUS slave device. In addition, the slave is informed that a write (DATO/DATOB back to the same address) will soon follow. In UNIBUS terminology, this cycle is called a DATIP (Data-In Pause).

This cycle originated in the days of magnetic core memory. Reading data from a core memory is a destructive process. That is, the very process of reading the data from the magnetic cores destroys the data, clearing it to zeroes (see Figure 5-18). If the core memory location is to retain its data, the read/destroy operation must immediately be followed by a "restore" operation to write the data back into the cores.

However, when the master device knows that some form of Read/Modify/Write operation is being performed (e.g., an *increment* instruction), there is no need to restore the old data to the cores. In fact, core memory requires that the cores be cleared to zero prior to writing new data into them. Rather than allow the automatic restoration of the data, the master indicates that it will shortly supply new data and that the core memory should pause, rather than restore the old data. Hence the name of the cycle.

DATIP/DATO or DATIP/DATOB requires about half the time compared to DATI/DATO or DATI/DATOB. See Figure 5-19.

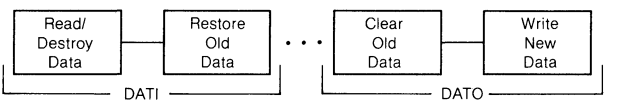


Figure 5-18 ▪ Separate Read and Write Cycles

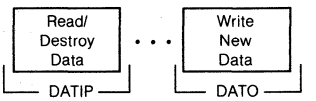


Figure 5-19 ▪ DATIP Followed by Write Cycle

Once the core memory has responded to the DATIP, it locks out all other operations except for the DATO/DATOB to the same location as read by the DATIP.

Write Vector

The final type of UNIBUS cycle has no mnemonic name. It is used by any master that desires to interrupt the interrupt-fielding processor (IFP). Figure 5-20 illustrates the case of a disk controller interrupting the IFP.

The master performs this cycle by using a special UNIBUS signal that always selects the IFP as the slave device and causes the master's data word to be written to the IFP. The IFP then takes the data word as the address of an interrupt service pointer (the *vector*) and interrupts through that vector.

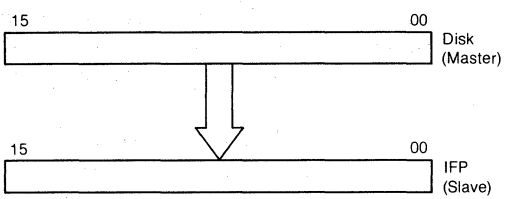


Figure 5-20 ▪ Interrupt Vector Write

By convention, the interrupt vector is in the range [000000-000774] but neither the UNIBUS hardware nor most IFPs impose any limit. If your master device passes 177774 as its vector, the IFP will try to interrupt through that vector.

Interrupt vectors are always located on even-word (quad-byte) boundaries. That is, they always take the form xxxxx0 or xxxxx4. IFPs will respond *unpredictably* to vectors of the form xxxxx2 or xxxxx6.

UNIBUS Signal Details

The UNIBUS actually consists of 56 signal lines and a number of ground (logic reference) lines. The individual lines are detailed in Table 5-1 and discussed in the subsequent paragraphs.

All lines on the bus employ open-collector drivers and resistive pull-ups.

Most of these signals are considered to be asserted (i.e., 1), while near ground, and are considered to be deasserted (i.e., 0) while near 3.3 volts. Signals of this type are indicated by the suffix L after the signal name. Any signal that is asserted while near 3.3 volts is indicated by the suffix H.

NOTE

All timing diagrams are drawn with up meaning asserted, rather than any particular voltage level.

Three different kinds of termination are used, depending on the particular signal line. The signals BUS AC LO L and BUS DC LO L are terminated by a resistor-capacitor network. In the table, this is indicated as "slow" termination. The grant lines (BUS NPG H, BUS BGx H) only connect from one module to the next and use a simplified termination on each module. In the table, this is indicated as "Grant" termination. All other lines are terminated at each end of the UNIBUS into 120 ohms and 3.3 volts. This is indicated as "fast" termination. Further details may be found under the heading "UNIBUS Electrical Characteristics."

The signals are partitioned into three groups which exactly follow the same grouping as previously described.

- Initialization and shutdown signals
- Arbitration signals
- Data transfer signals

In addition to those signals, which are actually part of the UNIBUS and controlled by the UNIBUS specifications, there are other standard signals which are present in most UNIBUS backplanes and follow UNIBUS-like conventions. These signals are documented under the heading of "Miscellaneous Signals."

Table 5-1 • UNIBUS Signals

Name	High/Low	Termination Type	Pin(s) In UNIBUS Cable	Pin(s) In SPC Backplane
• INITIALIZATION AND SHUTDOWN				
BUS DC LO L	L	Slow	BF2	CN1
BUS AC LO L	L	Slow	BF1	CV1
BUS INIT L	L	Fast	AA1	DL1

Name	High/Low	Termination Type	Pin(s) In UNIBUS Cable	Pin(s) In SPC Backplane
▪ ARBITRATION				
BUS NPR L	L	Fast	AS2	FJ1
BUS BR7 L	L	Fast	AT2	DD2
BUS BR6 L	L	Fast	AU2	DE2
BUS BR5 L	L	Fast	BC1	DF2
BUS BR4 L	L	Fast	BD2	DH2
BUS NPG H	H	Grant	AU1	In-CA1 Out-CB1
BUS BG7 H	H	Grant	AV1	In-DK2 Out-DL2
BUS BG6 H	H	Grant	BA1	In-DM2 Out-DN2
BUS BG5 H	H	Grant	BB1	In-DP2 Out-DR2
BUS BG4 H	H	Grant	BE2	In-DS2 Out-DT2
BUS SACK L	L	Fast	AR2	FT2
▪ DATA TRANSFERS				
BUS A00 L	L	Fast	BH2	EH2
BUS A01 L	L	Fast	BH1	EH1
BUS A02 L	L	Fast	BJ2	EF1
BUS A03 L	L	Fast	BJ1	EV2
BUS A04 L	L	Fast	BK2	EU2
BUS A05 L	L	Fast	BK1	EV1
BUS A06 L	L	Fast	BL2	EU1
BUS A07 L	L	Fast	BL1	EP2
BUS A08 L	L	Fast	BM2	EN2
BUS A09 L	L	Fast	BM1	ER1
BUS A10 L	L	Fast	BN2	EP1
BUS A11 L	L	Fast	BN1	EL1
BUS A12 L	L	Fast	BP2	EC1
BUS A13 L	L	Fast	BP1	EK2
BUS A14 L	L	Fast	BR2	EK1
BUS A15 L	L	Fast	BR1	ED2
BUS A16 L	L	Fast	BS2	EE2
BUS A17 L	L	Fast	BS1	ED1
BUS C0 L	L	Fast	BU2	EJ2
BUS C1 L	L	Fast	BT2	EF2
BUS D00 L	L	Fast	AC1	CS2
BUS D01 L	L	Fast	AD2	CR2

Name	High/Termination Low	Type	Pin(s) In UNIBUS Cable	SPC Backplane
BUS D02 L	L	Fast	AD1	CU2, FE2*
BUS D03 L	L	Fast	AE2	CT2, FL1*
BUS D04 L	L	Fast	AE1	CN2, FN2*
BUS D05 L	L	Fast	AF2	CP2, FF1*
BUS D06 L	L	Fast	AF1	CV2, FF2*
BUS D07 L	L	Fast	AH2	CM2, FH1*
BUS D08 L	L	Fast	AH1	CL2, FK1*
BUS D09 L	L	Fast	AJ2	CK2
BUS D10 L	L	Fast	AJ1	CJ2
BUS D11 L	L	Fast	AK2	CH1
BUS D12 L	L	Fast	AK1	CH2
BUS D13 L	L	Fast	AL2	CF2
BUS D14 L	L	Fast	AL1	CE2
BUS D15 L	L	Fast	AM2	CD2
BUS PA L	L	Fast	AM1	CC1
BUS PB L	L	Fast	AN2	CS1
BUS BBSY L	L	Fast	AP2	FD1
BUS MSYN L	L	Fast	BV1	EE1
BUS INTR L	L	Fast	AB1	FM1
BUS SSYN L	L	Fast	BU1	EJ1, FC1*
▪ MISCELLANEOUS				
BOOT ENABL L	L	N/A	<none>	AR1
HALT REQ L	L	N/A	<none>	CP1
HALT GRANT L	L	N/A	<none>	CR1
LTC	N/A	N/A	<none>	CD1

Initialization and Shutdown Signals

BUS DC LO L This signal, when asserted, indicates that somewhere on the UNIBUS the ac power has failed for sufficiently long that all other bus signals are not to be trusted. Each power supply in the system can drive BUS DC LO L.

This signal may also be driven by any UNIBUS device which desires to restart the system as though power had just been applied. Network interfaces use this feature to cause down-line loading to begin.

*For forwards compatibility, use the first pin rather than the second.

BUS AC LO L	This signal, when asserted, indicates that somewhere on the UNIBUS ac power has failed and the system will soon be inoperative. Upon the assertion of BUS AC LO L, the system processor will usually begin an orderly shutdown. Each power supply in the system can drive BUS AC LO L. Note that this signal (and BUS DC LO L, as well) is usually derived from each power supply's bulk dc source, not from direct measurements of the ac power line. This means that a lightly loaded power supply may not assert BUS AC LO L until quite awhile after the actual line power fails. This helps the processor ride through line transients but means that a system that must actually monitor the quality of the ac power line must not depend on BUS AC LO L. Other devices in the system may become inoperative prior to the assertion of BUS AC LO L. This signal may also be driven by any UNIBUS device which would like to cause the system to perform its power-fail or power-recovery action. This is also used by network interfaces.
BUS INIT L	This signal, when asserted, indicates that all devices on the UNIBUS should reset themselves to the state they entered upon initial powerup. BUS INIT L is asserted each time the system is started, by the PDP-11 RESET instruction, and by specific commands to the VAX-11 UNIBUS adapter or interface.

Powerup Timing

Powerup timing is quite variable depending on the specific power supplies and devices connected to the UNIBUS. A typical powerup sequence is shown in Figure 5-21. As ac power is applied to the system, the output(s) from each power supply becomes stable. As this occurs, each power supply releases BUS DC LO L. When the last power supply (stabilizes and) releases BUS DC LO L, BUS DC LO L is deasserted on the UNIBUS. At this point, certain devices (such as the system central processor) begin their initialization.

As the +5 Vdc power supply voltage to the processor(s) stabilizes, the processor(s) will assert BUS INIT L. BUS INIT L will remain asserted for between 10 microseconds and 120 milliseconds. The assertion of BUS INIT L will initialize the remaining devices on the UNIBUS.

As with BUS DC LO L, each power supply also drives BUS AC LO L. When the last power supply releases BUS AC LO L, BUS AC LO L is deasserted on the UNIBUS. When BUS INIT L deasserts, the processor(s) begins monitoring BUS AC LO L (which very likely has already deasserted). Once each processor detects that BUS AC LO L has deasserted, the processor begins operating as

directed by the user. This may mean that the operating system is bootstrapped, or it may mean that a recovery from a power failure is attempted. In any case, the system is now on the air.

In a system with a single power supply, once BUS AC LO L deasserts, it is guaranteed to remain deasserted for a minimum of 2 milliseconds. That is to say, the system guarantees that at least 2 milliseconds of time is available to recover from the previous power failure before the next power failure is signaled. In systems with more than one power supply, this guarantee cannot be made.

BOOT ENABL L is valid only at the instant that BUS DC LO L deasserts. Any device requiring this signal must latch it at this time.

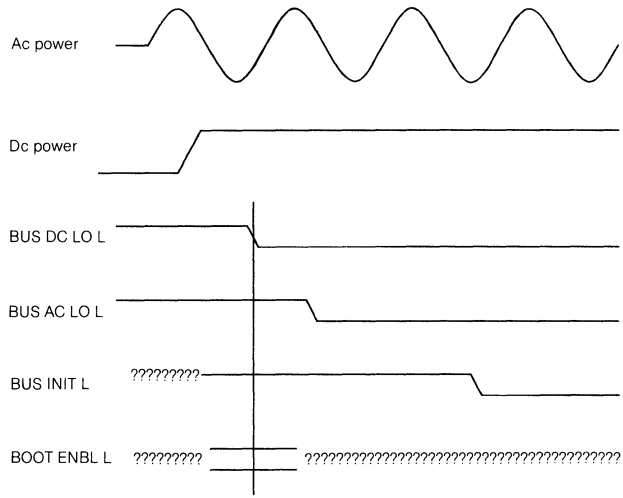


Figure 5-21 ▪ Typical Powerup Timing

Powerdown Timing

Powerdown timing is generally the reverse of powerup timing. It is also quite variable depending on the specific power supplies and devices connected to the UNIBUS. A typical powerdown sequence is shown in Figure 5-22.

Once ac power fails, the storage capacitors in the various power supplies begin discharging. As each power supply recognizes that it will soon have insufficient power to continue operation, it asserts BUS AC LO L. Effectively, BUS AC LO L is asserted by the first power supply to detect the failure. PDP-11 and VAX-11 central processors are notified by means of an interrupt that power is failing. A minimum of 2 milliseconds (5 milliseconds nominal) of runtime remains.

As each power supply discharges further, it asserts BUS DC LO L. Once BUS DC LO L has been asserted by any power supply, all activity in the system is halted. Each of the dc supply voltages actually remains valid for at least 5 milliseconds after the assertion of BUS DC LO L. BUS INIT L also asserts briefly when BUS DC LO L asserts.

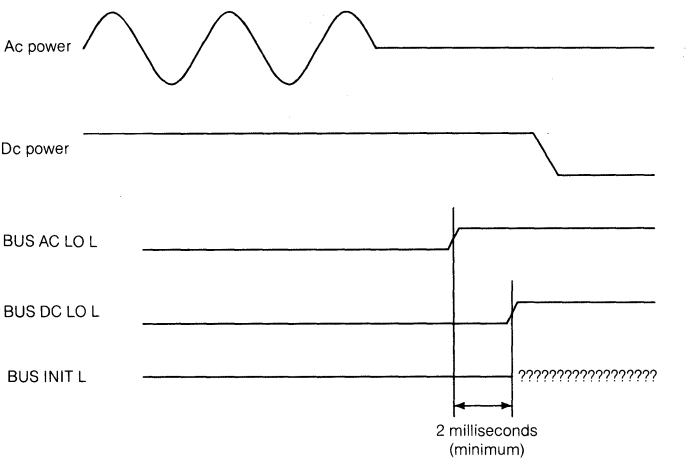


Figure 5-22 ▪ Typical Powerdown Timing

Initialization Timing

Any time the PDP-11 executes a RESET instruction, or the VAX-11 UNIBUS adapter or interface is commanded to reset the UNIBUS, BUS INIT L will be asserted. This signal resets most devices to the state they assumed immediately after powerup.

The BUS INIT L signal is a pulse. Its normal duration varies from 10 microseconds in the PDP-11/84 to about 120 milliseconds in the PDP-11/24. Typical timing is illustrated in Figure 5-23. In order to allow powerfail routines to complete, BUS INIT L may be prematurely terminated by the assertion of BUS AC LO L. In this case, a minimum assertion of BUS INIT L for about 1 microsecond is guaranteed by most PDP-11 processors and VAX-11 UNIBUS adapters/interfaces. The PDP-11/15, 11/20, 11/35, 11/40, and 11/60 do not terminate BUS INIT L upon a powerfail.

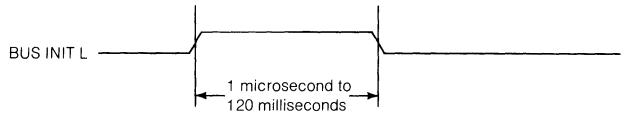


Figure 5-23 • UNIBUS Initialization Timing

Arbitration Signals

BUS BR4 L BUS BR5 L BUS BR6 L BUS BR7 L	These signals request that the Interrupt Fielding Processor (IFP) (usually the central processor) be interrupted at priority level 4, 5, 6, or 7, respectively. Priority 7 is the highest priority. If the IFP is running below the priority of the request, the interrupt will be granted at the end of the current instruction. If the IFP is running at or above the priority of the request, the request will be deferred until the processor priority is reduced.
BUS BG4 H BUS BG5 H BUS BG6 H BUS BG7 H	These signals indicate that the IFP is willing to grant the interrupt at priority level 4, 5, 6, or 7, respectively. The interrupting device must now become bus master and pass its interrupt service vector address to the IFP. **NOTE** These signals are asserted high. They are also not used as wired-OR signals although they are driven with the same type of integrated circuits as all other lines. They are terminated in a slightly different manner than other UNIBUS signals.
BUS NPR L	This signal is used to request use of the UNIBUS (without causing an interrupt of the IFP). This signal is roughly analagous to the direct memory access (DMA) request signal defined by other buses.
BUS NPG H	This signal indicates that the bus arbitrator is willing to allow use of the UNIBUS by one of the devices asserting BUS NPR L. **NOTE** This signal, like the BUS BGx H signals, is aserted high. It is also not used as a wired-OR signal although it is driven with the same type of integrated circuits as all other lines. Also like the BUS BGx H signals, it is terminated in a slightly different manner than other UNIBUS signals.
BUS SACK L	This signal is asserted by a device that has accepted a grant (whether a BG or NPG). This signal means selection acknowledged. As long as BUS SACK L is asserted, further arbitration is stopped. This assures that only one device is designated as the next master of the UNIBUS.

UNIBUS Arbitration Timing

The sequence a device uses to arbitrate for use of the UNIBUS data transfer bus is shown in Figure 5-24 and described below.

1. Assert Request
   Our device, wishing to become the bus master, asserts a request line. This request may be ORed with other requests already present on the BRx/NPR line.

2. Receive Grant
   Sometime later, when our device's NPR or BRx is the highest-priority request presented to the bus arbitrator, the arbitrator will issue NPG, or, with authorization from the the IFP, BGx. If no higher priority device at level BRx or NPR wants the grant, the grant will be propagated to our device. Our device will receive the grant and not propagate it (i.e., we block the grant).

3. Assert SACK
   As soon as our device sees the grant, it asserts BUS SACK L to indicate that it acknowledges its selection as the next bus master.

4a. Remove Request
   Once BUS SACK L has been asserted, our device may remove its request.

4b. Removal of the Grant
   Once the arbitrator sees our device's assertion of BUS SACK L, the arbitrator will remove the grant. No further grants will be issued to anyone as long as our device holds BUS SACK L.

5. Data Cycles
   One or more data cycles follow, and are described later.

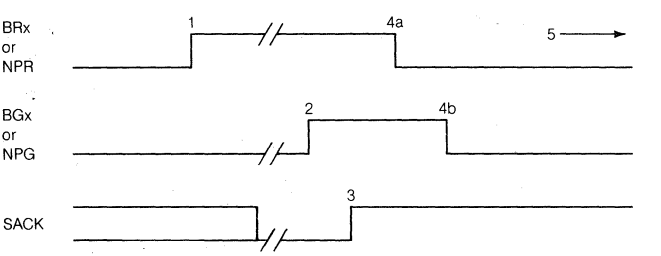


Figure 5-24 ▪ Arbitration Timing

Abnormal Cycles

Five abnormal arbitration cycles exist.

- Grant refusal
- No-SACK timeout
- Passive release
- Stolen grant (good guy interrupter)
- Stolen grant (bad guy DMA)

- GRANT REFUSAL

After making a request and being given a grant, our device may choose not to take the grant. In this case, the grant passes through our device. If another device further down the bus is waiting for the grant, the arbitration cycle continues normally (from the point of view of the device down the bus). If no device on a particular BRx/NPR level takes the grant, the grant reaches the bus terminator. Some UNIBUS terminators do nothing with the grant—this leads to the no-SACK timeout described below. Other UNIBUS terminators (notably the M9302) take all grants and reply with BUS SACK L. Then, once the grant is removed by the arbitrator, BUS SACK L will be deasserted by the terminator and a passive release occurs.

- NO-SACK TIMEOUT

The arbitrator only issues one grant at a time, and the UNIBUS is considered to be busy while a grant is being issued. This means that if a grant is issued, and no one asserts BUS SACK L, the system will hang waiting for its assertion. This situation often occurs if noise is being coupled onto the request lines. In this case, the arbitrator issues a grant in response to the noise but there are no genuine requesters on the UNIBUS. All PDP-11s and VAX-11 UNIBUS adapters and interfaces therefore contain a feature called No-SACK Timeout. If BUS SACK L is not received within 10–25 microseconds of issuing a grant, the arbitrator removes the grant and a passive release occurs. This prevents the system from hanging while waiting for a grant.

- PASSIVE RELEASE

At any time, the device selected to be the next bus master can say "nevermind" by just releasing BUS SACK L. Because no actual data cycles occurred, this is referred to as a passive release of the UNIBUS. Passive releases are harmless but should be avoided because they waste time.

- STOLEN GRANT (GOOD GUY INTERRUPTER)

The UNIBUS treats NPR as higher in priority than any of the BRx interrupt requests. However, once any grant is issued, the master granted use of the

UNIBUS is allowed to take as much time as needed. In addition, certain older central processors (e.g., the original PDP-11) did not allow an NPR to interleave with any (and all) processor operations. Rather, NPRs, like BR interrupt requests, were only granted at the end of processor instructions (or similar operations).

Certain older DMA peripheral devices (e.g., TC11 and RK11-C) contain very little buffering, and must be granted immediate use of the UNIBUS once they require that a word be transferred. If the arbitrator has just issued a BG (to an interrupting device), the total time required for the processor (the IFP) to get the vector, save the PC and PS, and load the PC and PS from the vector was too long for the DMA device to wait, and a data-late error occurred in the device.

To alleviate this problem, some old interrupting devices incorporate a feature called Steal Grant. If a BUS BGx H is propagating down the UNIBUS and reaches one of these devices, the device examines BUS NPR L. If BUS NPR L is asserted (meaning that an NPR device is waiting), the device blocks the BGx grant, and asserts BUS SACK L. In effect, it has stolen the BGx intended for some device on the UNIBUS. Once the arbitrator sees the assertion of BUS SACK L, it deasserts BUS BGx H. The stealing device then deasserts BUS SACK L (a passive release!). The arbitrator re-arbitrates and now issues the desired BUS NPG H in response to BUS NPR L.

As processors became more flexible about granting NPRs, and DMA devices have come to contain more buffering, this feature has fallen into disuse. On most Digital modules, the steal grant feature is controlled by a jumper named N1.

- STOLEN GRANT (BAD GUY DMA)

  While an interrupting device can never issue BUS INTR L while using the bus under the authority granted by BUS NPG H, it is perfectly legal for a device to become bus master under the authority of BUS BGx H and then do a number of DMA bus cycles prior to performing the vector write cycle. Stretching this just a little suggests that it is perfectly acceptable for a DMA device to borrow the BUS BGx H intended for an interrupt device, do a few bus cycles, and then pass BUS BGx H on to the interrupting device. The interrupting device generally won't know that its authority to become next bus master has been borrowed. The only possible UNIBUS failure that can occur is a no-SACK timeout should the bad guy DMA device keep the grant too long. Bad guy DMA has never been used by a Digital device.

Data Transfer Signals

BUS BBSY L	Also known as BUS BUSY L, this signal indicates that the UNIBUS data transfer bus is occupied (by the current bus master). The next bus master monitors this signal to determine when it can seize the data transfer bus.
BUS A17 L through BUS A00 L	These 18 signals are referred to as the UNIBUS address lines and supply the address that selects a slave device. BUS A00 L is the least significant bit and is ignored for all cycles except DATOB (write byte).
BUS C1 L BUS C0 L	These two signals are referred to as the UNIBUS control lines and designate four out of the five types of operations of the data transfer bus. The operations are described in detail later in this chapter.

C1	C0	Cycle Type
0	0	DATI (Read word)
0	1	DATIP (Read with intent to modify)
1	0	DATO (Write word)
1	1	DATOB (Write byte)
x	x	WRITE VECTOR (see BUS INTR L)

BUS D15 L through BUS D00 L	These 16 signals are referred to as the UNIBUS data lines and transmit the data word. BUS D00 L is the least significant bit. Byte data is *not always transmitted on* D07-D00. While the data for bytes at even addresses (A00=0) is transmitted on D07-D00, data for bytes at odd addresses (A00=1) is transmitted on D15-D08. This generally simplifies the design of word-oriented devices that may also read and write bytes.
BUS PA L BUS PB L	These two signals are used to convey parity error information between the master and slave devices (e.g., memories assert BUS PB L to indicate that a parity error was detected within the memory).

PA	PB	Status
0	0	Good parity detected within slave
0	1	Bad Parity detected within slave
1	X	Master should ignore PA (i.e., parity detection is disabled by someone on the UNIBUS). This combination is not used by Digital

BUS MSYN L	This signal serves two purposes. As the UNIBUS address strobe, it indicates that the UNIBUS address lines contain a valid address and that BUS C1 L and BUS C0 L contain a valid function code. For write word and write byte operations (DATO and DATOB), BUS MSYN L also serves as the data strobe, indicating that the master has placed valid data on the UNIBUS data lines.
BUS SSYN L	This signal also serves two functions. As the UNIBUS address acknowledgement, it serves to inform the bus master that the requested address exists within some slave. For read operations (DATI and DATIP), BUS SSYN L also serves as the data strobe, indicating that the slave has placed valid data on the UNIBUS data lines and valid parity status information on BUS PA L and BUS PB L.
BUS INTR L	This signal provides for a fifth kind of UNIBUS data cycle known as WRITE VECTOR. BUS INTR L acts simultaneously as the address, the address strobe, and the data strobe. The IFP is always the selected slave. The contents of the UNIBUS address lines, BUS C1 L, and BUS C0 L are ignored. The contents of the UNIBUS data lines are accepted by the IFP as an interrupt vector. The IFP then returns BUS SSYN L, just like any other bus slave.

Strobe	This is not a UNIBUS signal. It is typical of signals that exist within real devices. The signal is shown as a pulse indicating the moment that data should be latched into a device's flip-flops.

DATI Timing (Master Reads from Slave)

Assuming that our device has won the priority arbitration and asserted BUS SACK L, we can now proceed to observe a data transfer cycle. Please refer to Figure 5-25.

1. Receive the deassertion of the previous BBSY.
   Our device begins monitoring the BUS BUSY L line and waits until it sees BUS BBSY L unasserted.

2a. Assert BBSY.
   As soon as our device sees that BUS BBSY L is unasserted (indicating that the previous master is no longer using the data section of the UNIBUS), our device asserts BUS BBSY L. We are now the bus master.

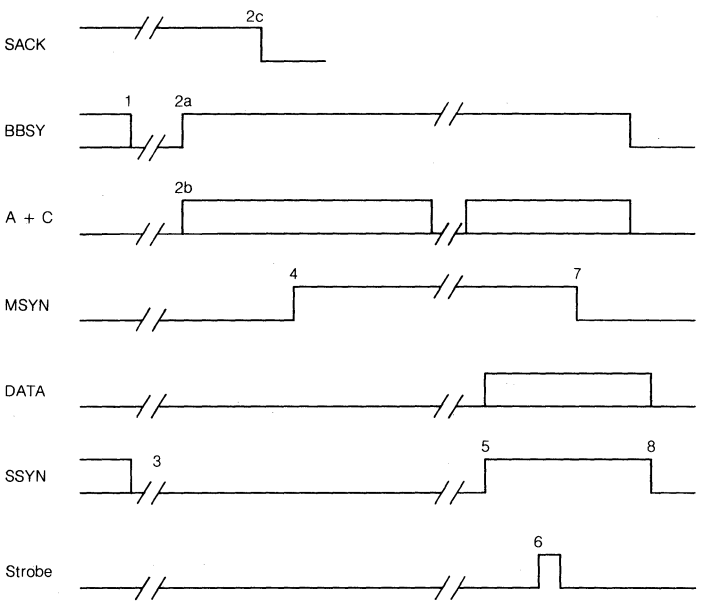


Figure 5-25 ▪ DATI/DATIP Timing

2b. Assert Address and Control
At the same time that we drive BUS BBSY L, we may also drive the UNIBUS address lines along with control information, BUS C1 L and BUS C0 L. All other devices on the bus immediately begin decoding this information to determine whether they will be the selected slave. A minimum of 150 nanoseconds must elapse prior to the assertion of BUS MSYN L. This time period allows the addressing information time to reach all points on the bus and be decoded. It is referred to as the front end deskew.

2c. Deassert SACK
Once our device asserts BUS BBSY L, it has become the bus master. If we are only transferring a single word, our device can now allow arbitration to resume. This is done by deasserting BUS SACK L. This may be done anytime after the assertion of BUS BBSY L, but should be done as early in the cycle as practical.

3. Receive the deassertion of the previous SSYN.
Any device on the UNIBUS is allowed to stall the onset of the next bus cycle by holding BUS SSYN L asserted. Our new bus master must ensure that BUS SSYN L is deasserted prior to proceeding. See paragraph 8.

4. Assert MSYN.
Once the 150 nanoseconds of front-end deskew time have elapsed, and any previous BUS SSYN L has been deasserted, our master may now assert BUS MSYN L. This informs everyone on the bus that the addressing information is valid.

5. Receive the assertion of SSYN
Our master device now waits for the selected slave to place the requested data onto the UNIBUS data lines (along with parity information on BUS PA L and BUS PB L). When the slave does this, it also asserts BUS SSYN L.

6. Deskew the Data
Our master must now allow 75 nanoseconds from the receipt of BUS SSYN L to guarantee that the returned data has all arrived at the master. This is referred to as the data deskew. Once this time has elapsed, the master may strobe in the returned data. Note that Strobe is not a bus signal but rather a typical signal within our master device.

7. Deassert MSYN
Once the master has captured the returned data, BUS MSYN L may be removed. The master must continue to hold the addressing information valid for at least 75 nanoseconds to ensure that all devices are notified that the address is invalid prior to it actually becoming invalid. This period is referred to as tail end deskew.

8. Receive the deassertion of SSYN
   Once the selected slave sees BUS MSYN L deassert, the slave will eventually deassert BUS SSYN L. The slave is completely within its rights to hold BUS SSYN L asserted for as long as necessary to complete whatever operations it must. For hardwired devices, BUS SSYN L usually is deasserted very quickly, but for microcoded devices, there may be a delay. Other devices (which are neither the current master nor slave) may also stretch BUS SSYN L once it appears. This capability allows cache memories and bus monitors adequate time to perform their business.

DATIP (Master Reads from Slave, No Restore)

DATIP (Data-In, Pause) is used when the master wishes to read a value from the slave but also expects to write back a new value.

DATIP differs from DATI only in the fact that the control line BUS C0 L is asserted. Most masters also keep BUS SACK L and BUS BBSY L asserted between the DATIP and the subsequent DATO or DATOB. This saves the time that would be spent rearbitrating for the UNIBUS. In the case of interlocked slaves (like the core memory), this also prevents other devices from becoming bus master and subsequently from being frustrated as they try to access the unresponsive memory.

Most I/O devices treat DATIP identically to DATI.

DATO/DATOB (Master Writes to Slave)

In many respects, a DATO/DATOB cycle is similar to a DATI cycle. The principal difference is that the master supplies the data to the slave. Please refer to Figure 5-26. The following paragraphs document only the differences between a DATI/DATIP and a DATO/DATOB. All other comments from DATI/DATIP apply to DATO/DATOB as well.

2b. Assert Address, Control, and Data.

   During a write cycle, we assert not only UNIBUS address and control information, but data as well. Note that the 150 nanoseconds of front end deskew is more than adequate to cover the 75 nanoseconds of required data deskew time.

4. Assert MSYN.
   Once the 150 nanoseconds of front end deskew has elapsed and any previous BUS SSYN L has deasserted, our master may now assert BUS MSYN L. This informs everyone on the bus that the addressing information and data is valid.

5. Receive the assertion of SSYN.
   Our master device now waits for the selected slave to accept the data it has presented on the UNIBUS data lines. When the slave does this, the slave will assert BUS SSYN L.

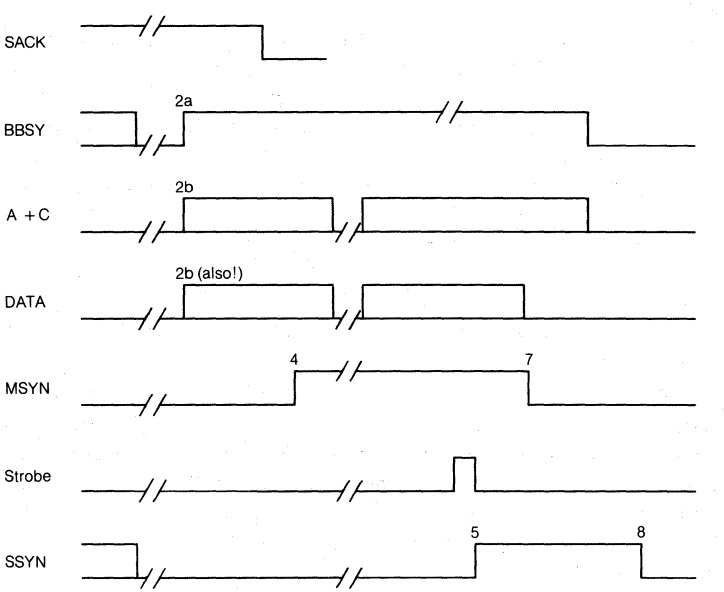


Figure 5-26 ▪ DATO/DATOB Cycles

7. Deassert MSYN.
   Upon receiving the returned BUS SSYN L, the master removes BUS MSYN L. The master must hold the addressing information valid for 75 nanoseconds to ensure that all devices are notified that the address is invalid prior to it actually becoming invalid (tail end deskew). The UNIBUS specifications don't require that the data also be held, but it certainly helps add margin to the system if you can do it.

Write Vector Timing (Master Interrupts IFP)

For the write vector operation, BUS INTR L replaces the functions of the UNIBUS address and control lines and BUS MSYN L. Please refer to Figure 5-27.

1. Receive the deassertion of the previous BBSY.
   Our device begins monitoring the BUS BUSY L line and waits until it sees BUS BBSY L unasserted.

2a. Assert BBSY.
   As soon as our device sees that the previous master is no longer using the data section of the UNIBUS, our device asserts BUS BBSY L. Our device is now the bus master.

2b. Deassert SACK.
Once our device asserts BUS BBSY L, it has become the bus master and can now allow arbitration to resume. This is done by deasserting BUS SACK L.

3. Receive the deassertion of the previous SSYN.
Any device on the UNIBUS is allowed to stall the onset of the next bus cycle by holding BUS SSYN L asserted. Our new bus master must ensure that BUS SSYN L is deasserted prior to proceeding.

4. Assert D15-D00 and INTR.
Once BUS SSYN L is deasserted, our device can immediately drive its interrupt vector onto the UNIBUS data lines and *simultaneously* drive BUS INTR L. This is *the one case where the master is not responsible for the deskewing.*

5. Receive the assertion of SSYN
The IFP, acting as the slave device, receives BUS INTR L and waits 75 nanoseconds to allow for the deskewing of the data (the interrupt vector). The IFP then strobes in the vector and returns BUS SSYN L. Meanwhile, as in any write cycle, our master device has been waiting for the slave to assert BUS SSYN L. Most masters do not contain the logic to timeout if the IFP fails to return BUS SSYN L.

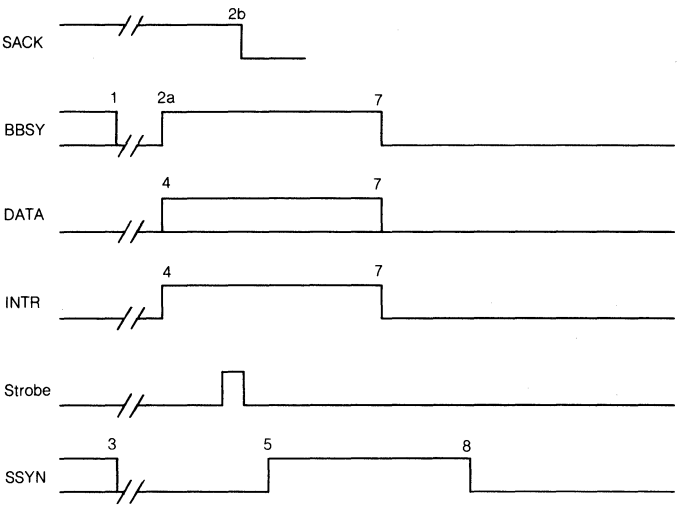


Figure 5-27 ▪ Write Vector Cycle

7. Deassert D15-D00, INTR, and BBSY
   After the IFP asserts BUS SSYN L, the master may deassert BUS INTR L and remove the vector from the UNIBUS data lines. Note that the IFP must really have meant it when it returned BUS SSYN L because the master is not obliged to provide any tail end deskew of the vector data.
8. Receive the deassertion of SSYN
   Once the IFP sees BUS INTR L deassert, it will eventually deassert BUS SSYN L.

Abnormal Bus Cycles

Aside from the abnormalities of arbitration, only one exception can occur.

- SSYN TIMEOUT

When an address is placed on the bus, and BUS MSYN L is asserted, the master device begins waiting for BUS SSYN L to be asserted by some slave. If no device exists at the specified address, BUS SSYN L will never be asserted. As described thus far, this situation would hang the bus.

To avoid this problem, it is the responsibility of the current bus master to monitor the amount of time that it has spent waiting for the assertion of BUS SSYN L. If too much time expires, the master must end the current bus cycle (using valid tail end timing). The bus master may then handle the error in whatever manner the designer sees fit. PDP-11 processors trap through the vector at 4 when a UNIBUS timeout error occurs. VAX-11 processors report the error in varying ways. Most I/O devices set a status bit and begin an error interrupt.

The SSYN timeout value varies from device to device but generally ranges from 10 to 25 microseconds. If the system uses multiported memory, it may be necessary to increase the timeout value. If a one-shot is used as the SSYN timer, you can simply increase the size of the capacitor. If a counter is used to provide the timeout, a variety of settings should be provided.

The unqualified phrase "bus timeout" usually refers to the specific case of SSYN timeout (rather than any of the other timeouts discussed in this chapter).

Multiple Bus Cycles (Burst Mode)

Once a device has gained UNIBUS mastership, it may do a single bus cycle and then release the UNIBUS, or it may do a series of UNIBUS data cycles while holding BUS BBSY L constantly asserted. Remember, once you gain ownership of the UNIBUS, it is yours without limit until you release BUS BBSY L.

If the device gained control of the UNIBUS under the authority of a BGx, it may perform a write vector as its last cycle.

Performing multiple data cycles during one UNIBUS mastership is generally referred to as burst mode. Usually, a small number of data cycles (1, 2, 4, sometimes 8 or 16) are performed. If the device performs an unlimited number of data cycles, it is referred to as a *bus hog* because this prevents other devices (including the CPU) from using the UNIBUS.

When a device performs multiple data cycles, it is usually best to keep BUS SACK L asserted until the last data cycle is underway. This ensures the fairest arbitration of the UNIBUS (because the priority arbitration will be made based on the most recent requests, not those many microseconds old). Refer to Figure 5-28.

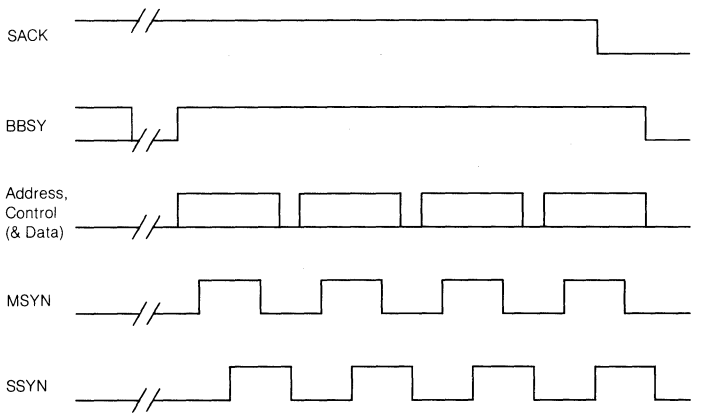


Figure 5-28 • Burst Mode (Showing Four-Cycle Burst)

Miscellaneous Signals

These signals are not part of the UNIBUS but are usually used by UNIBUS devices. They are present in most (but not all) UNIBUS backplanes.

BOOT ENABL L	This signal is derived from the power supply and indicates whether the system can reasonably expect the contents of a volatile memory (e.g., MOS) to be valid. If so (e.g., a battery backup unit maintained the contents of memory through a power failure), the signal is unasserted. However, if the contents of the volatile memory are not to be trusted (e.g., the battery went dead), then the signal is asserted, and the central processor is thus made aware that the operating software must be reloaded (bootstrapped). BOOT ENABL L is valid only at the instant that BUS DC LO L deasserts.

HALT REQ L	This signal appears in the CPU backplanes of the PDP-11/04, 11/24, 11/34, and 11/44. When asserted, it causes the processor to halt at the completion of the current instruction. *This signal is not available in any other backplane in the system.*
HALT GRANT L	This signal is issued by the PDP-11/04, 11/24, 11/34, and 11/44 when the CPU halts in response to HALT REQ L. *This signal is not available in any other backplane in the system.*
LTC	This signal is a square wave driven from the power supply at the ac power frequency (i.e., 60 Hz or 50 Hz). Since this signal has a nominal 50% duty cycle, and is not related in time to any other bus signal, it has no polarity indicator and may be construed as either logic high or logic low. LTC normally has slow rise and fall times (requiring the use of Schmitt-trigger receivers) and cannot drive as many receivers as other bus signals.

▪ UNIBUS Electrical Characteristics

The UNIBUS has the following major electrical characteristics.

- 56 signal lines, combined with grounds as necessary
- 120-ohm impedance
- Open-collector drivers
- Low-leakage receivers
- Termination at each end

  Thévenin voltage = 3.3 volts
  Thévenin resistance = 120 ohms

The 56 signal lines have already been described. Electrically, they fall into three major classes.

- Electrically bidirectional high-speed lines
- Electrically unidirectional high-speed lines
- BUS AC LO L and BUS DC LO L

Electrically Bidirectional High-speed Lines

Most of the lines on the UNIBUS are electrically bidirectional, although they may be logically unidirectional (e.g., BRx and NPR only carry signals *to* the bus arbitrator). A typical line is illustrated in Figure 5-29. All these lines

- Are terminated at each end of the UNIBUS into 120 ohms

  180 ohms to +5Vdc, 390 ohms to ground.
- Contain at least one open-collector driver.
- Contain at least one low-leakage receiver.
- Are asserted by pulling the line to ground (i.e., they are logic low lines).

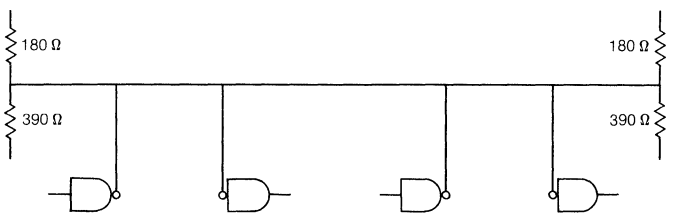


Figure 5-29 • An Electrically Bidirectional Line

Electrically Unidirectional High-speed Lines

The UNIBUS grant lines (BUS NPG H, BG7 H, BG6 H, BG5 H, and BG4 H) are electrically unidirectional (i.e., BGx and NPG only carry signals *from* the bus arbitrator). A typical segment of a grant line is illustrated in Figure 5-30. All these lines

- Are daisychained from module to module.
- Are terminated at each module in the daisychain with a simplified terminator.

  Towards arbitrator—180 ohms to +5Vdc.
  Away from arbitrator—180 ohms to +5Vdc, 390 ohms to ground.
- Contain only one open-collector driver.
- Contain only one low-leakage receiver.
- Are *deasserted* by pulling the line to ground and asserted by allowing the line to float (i.e., they are logic high lines).

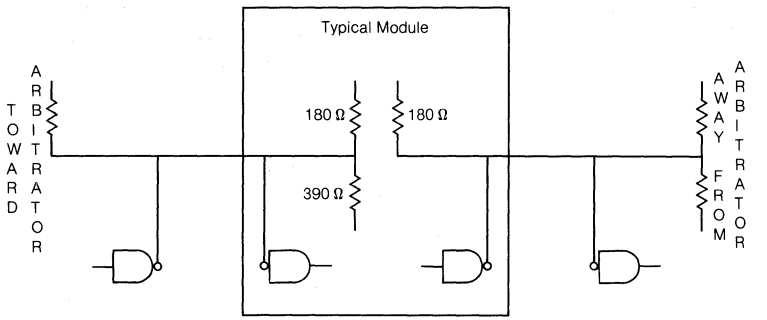


Figure 5-30 ▪ An Electrically Unidirectional Line

BUS AC LO L and BUS DC LO L

BUS AC LO L and BUS DC LO L greatly resemble the bidirectional lines. However, besides connecting to each device on the UNIBUS, connections from these two lines are also carried in wiring harnesses to each of the system's power supplies. This requires slow-down filtering to prevent noise from disrupting the operation of the system. Each power supply must drive the lines with some form of "normally on" drivers, so that the lines will be held grounded while the power supply is de-energized. This is often done with a depletion-mode FET (field effect transistor). Other devices use bias from the termination resistors to activate their drivers. A typical line is illustrated in Figure 5.31. Both BUS AC LO L and BUS DC LO L

- Are terminated at each end of the UNIBUS with a slow terminator

  390 ohms to +5Vdc.
  1000 picofarads to ground.

- Contain at least one open-collector or open-drain driver.
- Contain at least one low-leakage receiver.
- Are asserted by pulling the line to ground (i.e., they are logic low lines).
- Must remain asserted (grounded) when no other power is applied to the system.

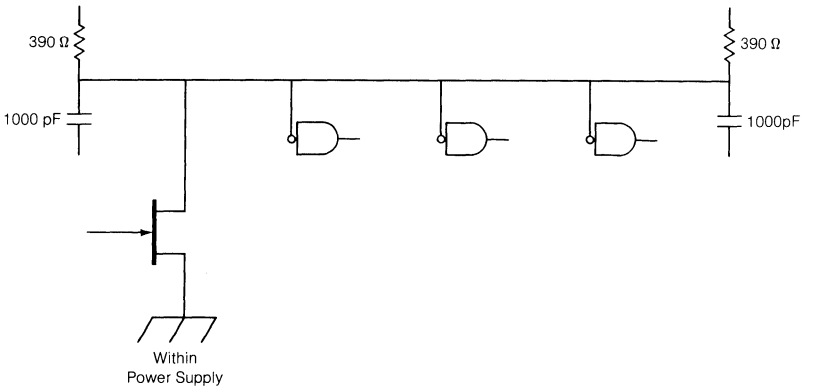


Figure 5-31 • BUS AC LO L or BUS DC LO L

• Design Suggestions

Some suggestions for UNIBUS designers are presented below. While not an exhaustive list, these suggestions cover the most common mistakes made while designing UNIBUS devices.

Watch the dc Voltage Levels

The UNIBUS depends on its voltage levels. These voltage levels can be maintained only if the driver and receivers operate optimally, the cable has a controlled resistance, and the ground reference at all points in the system is accurate.

- Use the correct drivers and receivers.

  Drivers must be able to pull down at least 60 milliamperes with a small Vol (0.6V or less).
  Drivers and receivers must have little leakage current.
  At the time of this printing, the most recent UNIBUS transceivers are Digital part number 19-14987.

- Establish a solid logic reference (ground) throughout the system. No driver or receiver can cope with more than a few millivolts of ground potential.

- Keep dc resistances low so as to minimize voltage drop.

- Limit any one UNIBUS segment to a total length of 15.25 meters (50 feet), including all cable segments, backplanes, and the hidden cable in the M9202 bus jumper modules. A segment of the UNIBUS is defined as that portion of the bus between the front- and back-end terminators (there can be more than one segment to a given UNIBUS).
- Limit any one UNIBUS segment to a total of 20 dc loads. A *dc load* is defined as one driver and one receiver, or one transceiver.
- If the UNIBUS must extend beyond 15.25 meters, or contain more than 20 dc loads, the one logical UNIBUS must be broken up into more than one UNIBUS segment, with a UNIBUS repeater between each segment. The UNIBUS repeater contains logic which amplifies the UNIBUS signals, making possible greater length and more bus loads, at the expense of transmission speed. Each of the individual UNIBUS segments between bus repeaters has its own termination.

Maintain 120-ohm Impedance and Watch ac Signals

The UNIBUS can operate correctly only if its signals are propagated quickly and cleanly from end to end. Signals propagate quickly and cleanly along a line only when the impedance of the line remains constant. Any time the signal encounters a different impedance, a small part of the signal is reflected. These reflections can accumulate and cause errors in the signal.

Each time a branch (or *stub*) is added to the bus (e.g., a module connected), capacitance is added to the bus. This additional capacitance is referred to as an *ac load,* since the capacitance only affects ac signals, not dc voltage levels. The capacitance also changes the local impedance of the UNIBUS.

If many modules are connected in a short length of the bus, these many small changes of bus impedance add up to a large change, and a large reflection can be generated. These many small capacitances lumped together are referred to as a *lumped load.*

- Minimize the distance between the UNIBUS and the drivers and receivers. Minimize the capacitance of the connecting etch run or wire.
- Keep lumped loads small and separate them with 120-ohm bus cable (as in the M9202 2-foot cable in a 1-inch module) or 120-ohm twisted pair (as in the DD11-DK).
- Terminate with accurate, pure resistances and accurate voltages.

Minimize Crosstalk

- Use drivers which generate edges with controlled rise and fall times.
- Be cautious routing UNIBUS signals. Route the critical timing signals (at a minimum, BUS SACK L, BUS BBSY L, BUS MSYN L, BUS INTR L, BUS SSYN L) away from the address and data lines.
- Don't allow any signals to glitch, *whether or not* such a glitch violates any timing specification. Glitches crosstalk much more readily than clean signals.
- Use twisted pair with an impedance of 120 ohms to route critical timing signals in backplanes.
- Maintain the 120-ohm impedance of the UNIBUS.

Consider All Timing Cases

Design your logic explicitly considering all of the UNIBUS operations, whether you intend to use them or not. For example, whether or not your device honors DATOB, consider what will happen if your device is requested to perform a DATOB. Don't ignore the case.

Remember to consider all of the data deskew and setup times. For devices acting as UNIBUS masters, the most important considerations are.

- The master must assert the UNIBUS address at least 150 nanoseconds prior to asserting BUS MSYN L (front-end deskew time).
- For read operations, the master must not strobe in the slave's data until at least 75 nanoseconds after the reception of BUS SSYN L at the master (data-deskew time for reads).
- For write word and write byte, the master must assert the UNIBUS data at least 75 nanoseconds prior to asserting BUS MSYN L (data-deskew time for writes).
- For write vector, the master must assert the UNIBUS data (the vector) at or before the assertion of BUS INTR L. This is the one case where the master is not responsible for the data deskew. Here, the IFP performs the deskew.
- The master must hold the UNIBUS address valid for at least 75 nanoseconds after deasserting BUS MSYN L (tail-end deskew).

Design is considerably easier for devices acting as UNIBUS slaves, since the master does most of the deskewing. The major considerations for slaves are

- For read operations, do not assert BUS SSYN L prior to asserting data.
- For write operations, do not assert BUS SSYN L prior to latching in the data.

- *Never use the falling edge of BUS MSYN L to latch the data* (i.e., don't use transparent latches to capture the data unless the latches are closed prior to or simultaneously with the assertion of BUS SSYN L).

Remember that the UNIBUS is asynchronous and requires careful attention to setup and hold times at flip-flops. Be particularly careful to ensure that all synchronizer circuits account for the possibility of metastability. This is particularly important around the logic that decides whether to pass or block UNIBUS grants, and around circuits that synchronize BUS SSYN L for use by synchronous logic.

Be A Good Bus Citizen

Design for speed—any time your device can save on the UNIBUS saves everyone time and increases the system's overall throughput.

- Your device should be ready to immediately read or write data once it becomes the bus master.
- During your device's last (and possible only) data cycle, drop BUS SACK L so that bus arbitration may resume.

If your device transfers many words at great speed (perhaps 100,000 words or more per second)

- Within your device, provide enough FIFO (First-In, First-Out) buffering to avoid data late errors should your device not win mastership of the UNIBUS for some time.
- Transfer two words per bus mastership. Better yet, let the user specify the burst size (but default to two words).
- Design in sufficient *high-speed* buffering at the bus interface so that the multiple words transfer as quickly as possible. Do not use a microprocessor to produce the words one at a time, at the microprocessor's speed.
- Allow the user some method of throttling the data rate (e.g., provide an adjustable delay between the time that your device releases the bus and the next time that it requests the bus).

Appendix A—PDP-11 Family Differences Table

The table that follows illustrates the issues involved in software migration between different members of the PDP-11 family. Each member of the family has some small difference in the way it executes instructions. Any program developed using PDP-11 operating systems with higher level languages will migrate from one system to another with very little difficulty. However, some applications written in assembly language may have to be modified slightly.

The table also details the slight differences between the various PDP-11 family processors and modules. Available hardware options and memories occasionally vary from processor to processor. Sometimes there is a small difference in the specific feature of one processor or another (e.g., a given feature may be standard on one module, optional on another, or not applicable). These variances are categorized under seven major headings—instructions or instruction sets, memory management expansion and relocation, interrupts, buses, general purpose registers, error handling, and consoles.

The VAX column refers to the PDP-11 Compatibility Mode available on VAX-11 processors.

Key to the table:

Y	—Yes
N	—No
▨	—Not applicable to that processor or module
Std	—Standard
Opt	—Optional
With "option"	—This feature is available only if the named option is available.

Instructions or Instruction sets

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
BASIC	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SOB, SXT	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	N	Y	N	Y	Y	Y	Y
RTT	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
MARK	Y	N	N	Y	Y	Y	Y	N	N	N	Y	Y	N	Y	N	Y	Y	Y	N
XOR	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
ASH, ASHC, MUL, DIV	With KEV11 or KEV11-B	N	N	Y	Y	Y	Y	N	N	N	Y	Y	With KE11-E	Y	Y	Y	Y	Y	Y
46 Floating-point instructions	N	N	N	With KEF11-A or FPF11	With KEF11-A or FPF11	Y	Y	N	N	N	With KEF11-A or FPF11	With FP11-A	N	With FP11-A	With FP11-B or FP11-C	Y	With FP11-B or FP11-C	Y	N
MFPT	N	Y	Y	Y	Y	Y	Y	N	N	N	Y	N	N	Y	N	N	N	Y	N
MTPS	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	N	N	N	N	N	Y	N
KE11-A, -B available for MUL, DIV, SHIFT	N	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
CIS	*	N	N	With KEF11-B	With KEF11-B	N	N	N	N	N	With KEF11-B	N	N	With KE44	N	N	N	N	N
MFPI, MFPD, MTPI, MTPD	N	N	N	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
SPL	N	N	N	N	N	Y	Y	N	N	N	N	N	N	Y	Y	N	Y	Y	N
CSM	N	N	N	N	N	Y	Y	N	N	N	N	N	N	Y	N	N	N	Y	N

*Limitea subset
(DIS) with KEV11-C

Instructions or Instruction sets

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Illegal Instruction Actions:																			
OPcodes 000100–000107 JMP %n (jump to register)	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_4	T_4	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_4	T_4	T_{10}	T_4	T_4	T Native
OPcode 000000 (halt) in supervisor or user mode				T_{10}	T_{10}	T_4	T_4				T_{10}	T_{10}	T_{10}	T_4	T_4	T_{10}	T_4	T_4	T Native
In kernel mode	Halt	$T_{pup}+4$	$T_{pup}+4$	Halt or T_{10}	Halt or T_{10}	Halt or T_{10}	Halt or T_{10}	Halt	Halt	Halt	Halt or T_{10}	Halt	Halt	Halt	Halt	Halt	Halt	Halt or T_{10}	T Native
OPcodes 075040–75377 OPcodes 075400–75777		T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T Native
OPcode 076600	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	Med	T_{10}	T_{10}	T Native
OPcodes 170000–177777	User Microcode or T_{10}	T_{10}	T_{10}	FPP or T_{10}	FPP or T_{10}	FPP	FPP	T_{10}	T_{10}	T_{10}	FPP or T_{10}	FPP or T_{10}	T_{10}	FPP or T_{10}	FPP or T_{10}	FPP	FPP or T_{10}	FPP	T Native
OPcodes 210–227	Maint. Instruc.	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T_{10}	T Native
JMP, JRS																			
For JMP or JRS Mode 2, PC will receive	R	R	R	R	R	R	R	R	R+2	R+2	R	R	R	R	R	R	R	R	R

Instructions or Instruction sets

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
MOV(B),CMP(B),BIT(B) BIC(B),BIS(B),ADD,SUB:																			
If an instruction uses a register as the source operand, and the same register in autoincrement, autodecrement, autoincrement-deferred, or autodecrement-deferred mode (e.g., MOV R,(R)+), the destination will receive	R	R±2	R±2	R±2	R±2	R±2	R±2	R	R	R±2	R±2	R	R±2	R	R	R±2	R	R±2	R
If an instruction uses the PC as the source operand and indexed or indexed-deferred mode for the destination operand (e.g., MOV PC, HERE), the destination will receive	PC	PC+2	PC+2	PC+2	PC+2	PC+2	PC+2	PC	PC	PC+2	PC+2	PC	PC+2	PC	PC	PC+2	PC	PC+2	PC
SWAB:																			
V-bit action	0	0	0	0	0	0	0	0	0	—	0	0	0	0	0	0	0	0	0
EIS Differences:																			
EIS exists	With KEV11	N	N	Y	Y	Y	Y	N	N	N	Y	Y	With KE11-E	Y	Y	Y	Y	Y	Y
If R = Rv1 (in other words, the register is odd and a 16-bit result is being produced), are the condition codes based on the 16-bit or 32-bit result?				32	32	32	32				32	16		16	32		32	32	

Instructions or Instruction sets

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
FIS Differences (floating-instruction set):																			
FIS exists	With KEV11	N	N	N	N	N	N	N	N	N	N	N	With KE11-F	N	N	N	N	N	N
FMUL and FDIV require one word of RB stack	Y												N						
Is interruptible	Y												N						
Condition codes after FIS is interrupted are indeterminate	Y																		
MFPT Differences:																			
MFPT exists	N	Y	Y	Y	Y	Y	Y	N	N	N	Y	N	N	Y	N	N	N	Y	N
MFPT result		4	4	3	3	5	5				3			1				5	
FPP Differences:																			
FPP microcode	N	N	N	With KEF11-A	With KEF11-A	Y	Y	N	N	N	With KEF11-A	N	N	N	N	Y	N	Y	N
FPP hardware	N	N	N	With FPF11	With FPF11	With FPJ11	With FPJ11	N	N	N	With FPF11	With FP11-A	N	With FP11-F	With FP11-B FA11-C	With FP11-E	With FP11-B FP11-C	Y	N
Hardware FPP is synchronous or asynchronous				Sync	Sync	Async	Async				Sync	Sync		Sync	FP11-B Async FP11-C Sync	Async	FP11-B Async FP11-C Sync	Async	
FPP instructions can be interrupted				Y with KEF11-A N with FPF11-A	Y with KEF11-A N with FPF11-A	N	N				Y with KEF11-A N with FPF11-A	N		N	N	N	N	N	

Instructions or Instruction sets

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Reset Instruction:																			
Nominal Duration of assertion of BUS INIT L	10μs + 90μs idle	8.4μs + 0.15μs idle	8.4μs + 0.15μs idle	10μs + 90μs idle	10μs + 90μs idle	15μs + 130μs idle	15μs + 130μs idle	25 ms	150 ms	20ms + 70ms idle	100 ms	100 ms	20ms + 70ms idle	100 ms	25 ms	10 ms	25 ms	15μs + 130μs idle	
Detection of power failure ends RESET instruction	N	N	N	N	N	N	N	Y	Y	N	Y	Y	N	Y	Y	N	Y	N	
Minimum assertion of BUS INIT L (as shortened by power failure)								1μs	300ns			1μs		1μs	1μs		1μs		
Power failure during fetch of RESET instruction is fatal: no power down is executed	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	
MTPS/MFPS:																			
MTPS/MFPS exists	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	N	N	N	N	N	Y	PSL only affected in native mode
While in user or supervisor mode, MTPS changes PSW ⟨3-0⟩	Y	Y	Y	Y	Y	Y	Y				Y	Only if 777776 mapped						Y	
While in user or supervisor mode, MTPS changes PSW ⟨7-4⟩	N	N	N	N	N	N	N				N	Only if 777776 mapped						N	
While in user or supervisor mode, MFPS accesses PSW ⟨7-0⟩	Y	Y	Y	Y	Y	Y	Y				Y	Only if 777776 mapped						Y	

Memory Management Expansion and Relocation

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Number of physical Address bits	16	16	16	16, 18, or 22	22	22	22	16	16	16	18 22	18	16 18 with KT11-D	22	16 18 with KT11-C	18	22	22	various
Maximum physical memory (assuming 8-Kbyte I/O Page)	56	56	56	56, 248, or 4088	4088	4088	4088	56	56	56	248 or 3840	248	56 248 with KT11-D	3840	56 248 with KT11-C	248	3840	3840	various
Kernel mode?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Supervisor mode?	N	N	N	N	N	Y	Y	N	N	N	N	N	N	Y	Y with KT11-C	N	Y	Y	N
User mode?	N	N	N	Y	Y	Y	Y	N	N	N	Y	Y	Y with KT11-D	Y	Y with KT11-C	Y	Y	Y	N
Instruction space	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Separate data space	N	N	N	N	N	Y	Y	N	N	N	N	N	N	Y	Y	N	Y	Y	N
Action on any access with PSW bits ⟨15-14⟩ (current mode field) set to:																			
00 Kernel mode				Kernel	Kernel	Kernel	Kernel				Kernel	Kernel	Kernel	Kernel	Kernel	Kernel	Kernel	Kernel	
01 Supervisor mode				User	User	Super-visor	Super-visor				User	T_{250}		Super-visor	Super-visor	Kernel	Super-visor	Super-visor	
10 Reserved				User	User	T_{250}	T_{250}				User	T_{250}		T_{250}	T_{250}		T_{250}	T_{250}	
11 User mode				User	User	User	User				User	User	User	User	User	User	User	User	
Which SP does MFPI, MFPD, MTDI, and MTPD use when PSW ⟨13-12⟩ = 10?				Unpredictable		User	User				◄		Unpredictable				►	User	
PSW⟨15-12⟩, multiple SPS, and MTPI, MFPI, MTPD, and MFPD exist with or without MMU option				Y	Y	Std	Std				Y	Std	N	Std	Y	Std	Std	Std	

Memory Management Expansion and Relocation

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	Various	DCT11	KXT11-A	KDF11-A	KDF11-B	KDJ11-A	KDJ11-B	KD11-D	KD11-B	KC11-KA11	KDF11-UA	KD11-E	KD11-A	KD11-Z	KD11-A KD11-D	KD11-K	KB11-B KB11-C	KDJ11-B	Various
MMU Registers																			
MMR0 〈08〉 (maintenance) implemented				N	N	N	N				N	Y	Y	Y	Y	Y	Y	N	
MMR0 〈12〉 MMU trap flag implemented				N	N	N	N				N	N	N	N	Y	N	Y	N	
MMR1 exists but always reads 000000				Y	Y	N	N				Y	N	N	N	N	N	N	N	
MMR1 exists and operates				N	N	Y	Y				N	N	N	Y	Y	N	Y	Y	
MMR2 tracks instruction fetches				Y	Y	Y	Y				Y	Y	Y	Y	Y	Y	Y	Y	
MMR2 tracks interrupt vectors				N	N	N	N				N	N	N	N	Y	N	Y	N	
MMR3: Bits 〈00-02〉 (ENABLES DSPACE) exists and functions				N	N	Y	Y				N	N	N	Y	Y	N	Y	Y	
Bit 〈03〉 (ENABLE) exists and functions				N	N	Y	Y				N	N	N	Y	N	N	N	Y	
Bit 〈04〉 (ENABLE 22-bit exists and functions					Y	Y	Y				Y	N	N	Y	N	N	Y	Y	
Bit 〈05〉 (ENABLE UB map) exists and functions				← Exists only →							With KT24	N	N	Y	N	N	Y	Y	
PARs:																			
Width (bits)				16	16	16	16				16	12	12	16	12	12	16	16	
Program can execute from PAR				Y	Y	N	N				Y	12-bits only	12-bits only	Y	12-bits only	12-bits only	Y	N	
Bit 〈00〉 (trap any access) implemented				N	N	N	N				N	N	N	N	Y	N	Y	N	
Bit 〈07〉 (any access flag) implemented				N	N	N	N				N	N	N	N	Y	N	Y	N	
Bit 〈15〉 (bypass cache) implemented				N	N	Y	Y				N	N	N	Y	N	N	N	Y	

Interrupts

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Number of interrupt (BR) levels	1	4		4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	
Does expected interrupt occur if PSW ⟨07-05⟩ are lowered for only one instruction?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	
Can an interrupt service routine itself be interrupted prior to executing its first instruction?	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	
Are instructions interruptible?																			
EIS	N			N	N	N	N				N	N	N	N	N	N	N	N	N
FIS	Y												N						
FPP				Y with KEF11-A	Y with KEF11-A	N	N				Y with KEF11-A	N		N	N	N	N	N	
CIS (DIS)	Y			Y	Y						Y			Y					
PIRQ (Programmed interrupt request register)																			
Exists	N	N	N	N	N	Y	Y	N	N	N	N	N	N	Y	Y	N	Y	Y	N
Is cleared by reset						Y	Y							N	Y		Y	Y	

Buses

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Buses Available:																			
Q-bus	Y	N	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N
UNIBUS	N	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	1-4
Special Memory Bus											EUB			EUB	Digital Fast Bus		Cache Bus	PMI	Various
Bus Cycles Utilized:																			
CLR, SXT, do only DATO for last bus cycle (alternative: DATIP-DATO)	N	N	N	Y	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	Varies
MOV does only DATO for last bus cycle	Y	DATI, DATO	DATI, DATO	Y	Y	Y	Y	DATIP, DATO	DATIP, DATO	DATIP, DATO	Y	Y	Y	Y	Y	Y	Y	Y	Varies
EIS does DATI to fetch source operand	DATIP, DATO	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Varies
UNIBUS/Q-bus timeout value	10μs		10μs	10μs	10μs	10μs	10μs	22μs	22μs	25μs	25μs	25μs	15μs	25μs	7μs	35μs	7μs	15μs	Various
NPRs (DMAs) will be granted during CPU instructions	Y		Y	Y	Y	Y	Y	Y	Y	With KH11	Y	Y	Y	Y	Y	Y	Y	Y	Y
Console SLU accessible from Q-bus/UNIBUS	Y		N	Y	N	Y	N	Y	N	Y	N	Y	Y	N	Y	Y	Y	N	N
Line clock register accessible from a Q-bus/UNIBUS	*		N	Y	N	Opt	N	Y	N	Y	N	Y	Y	N	Y	Y	Y	N	N
Bootstrap ROMs accessible from Q-bus/	Y		N	*	N	*	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	1	N

* If LTC/Bootstrap exists at all.
1 M9312 Emulation–Yes.
Internal ROMs–No.

Processor Status Word

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Bits ⟨15-12⟩ mechanized (Current mode, Previous mode)	N	N	N	Y	Y	Y	Y	N	N	N	Y	Y	With KT11-D	Y	Y	Y	Y	Y	
Bit ⟨11⟩ mechanized (Register set selection)	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	Y	N	Y	Y	
Bit ⟨08⟩ mechanized (CIS instruction suspended)	N	N	N	Y	Y	Y	Y	N	N	N	Y	N	N	Y	N	N	N	Y	
Bit ⟨07⟩ mechanized (High-order priority bit)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	

Processor Status Word

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Bits ⟨06-05⟩ mechanized (Low-order priority bits)	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
Bit ⟨04⟩ mechanized (T)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
Bits ⟨03-00⟩ mechanized (N, Z, V, and C)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Processor Status Word can be accessed via Reads/Writes to location 17 777 776	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
MTPS/MFPS instructions (Bits ⟨07-00⟩ or ⟨03-00⟩	Y	N	N	Y	Y	Y	Y	N	N	N	Y	Y	N	N	N	N	N	Y	N
SPL instruction (Bits ⟨07-05⟩	N	N	N	N	N	Y	Y	N	N	N	N	N	N	Y	Y	N	Y	Y	N
Condition code instructions (bits ⟨03-00⟩	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
T-bit Differences:																			
Can explicit PSW reference (by program or console) Set/Clear T-bit?	N	N	N	N	N	N	N	Y	Y	Y	N	N	N	N	N	N	N	N	N
Number of instructions executed between RTI setting T-bit and T-bit TRAP:	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
Number of instructions executed between RTT setting T-bit and T-bit trap	1	1	1	1	1	1	1	1	No RTT Instruction	No RTT Instruction	1	1	1	1	1	1	1	1	1
T-bit TRAP immediately ends WAIT instruction	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Y	
T-bit TRAPS have higher priority than interrupts (are taken prior to interrupts)				Y	Y	Y	Y		Y		Y	Y	Y		N		N	Y	

General Purpose Registers

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Number of sets of R0-R5	1	1	1	1	1	2	2	1	1	1	1	1	1	1	2	1	2	2	1
Number of stack pointers	1	1	1	3 (2 useful)		3	3	1	1	1 (2 useful)	3	2	2	3	3	2	3	3	1
Can program code be executed from GPRs?	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N
Can GPRs be accessed by program as 177777700-177777717?	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N
Can GPRs be accessed by console as 177777700-177777717?	N	N	N	N	N	N	N	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	N	N

Error Handling

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Odd Address/Bus Errors																			
Odd address errors detected by CPU	N	N	N	N	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y
If odd address error occurs while autoincrementing/ autodecrementing a register, register will have been modified.						Y	Y	N	N	N		N	Y	N	Y		Y	Y	
Bus error (timeouts) detected by CPU	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
If bus error occurs while using autoincremented/ autodecremented-mode addressing, register will have been modified.	Y			Y	Y	Y	Y	N	Y	Y	Y	N	Y	N	Y	Y	Y	Y	
If bus error occurs while reading I-stream using PC, PC will have been incremented.	Y			Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	N	

Error Handling

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Errors Using (SP):																			
Odd address/Bus error while using (SP)	HALT	No error detected		SP←4 T_4[1]	SP←4 T_4[1]	SP←4 T_4	SP←4 T_4	HALT	HALT	HALT	SP←4 T_4[1]	HALT	SP←4 T_4	HALT	SP←4 T_4	SP←4 T_4	SP←4 T_4	SP←4 T_4	
Stack Overflow Errors:																			
Yellow-zone trap implemented?	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
Yellow-zone trap programmable (else fixed at 000400)				N	N	N	N	N	N	N	N	N	With KJ11	N	Y	Y	Y	N	
Yellow-zone action				Execute then T_4	Execute then T_4	Execute then T_4	Execute then T_4	T_4	T_4		Execute then T_4	T_4	Execute then T_4	Execute then T_4	Execute then T_4	Execute then T_4	Execute then T_4	Execute then T_4	
Separate red-zone trap implemented?				N	N	N	N	N	N	N	N	N	Y	N	Y	Y	Y	N	
Red-zone action													T_4		SP←4 T_4	T_4	SP←4 T_4		
Vector Errors:																			
Error while fetching error, vector hangs processor.	Y	No error not detected		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
Halt switch halts processor.	N			N	N	Y	Y	N	N	N	N	N	N	Y	N	N	N	Y	
Console initialize halts processor.								Y	Y	Y		Y	Y	Y	Y		Y		
Console RESTART switch restarts processor.	Y			Y	Y	Y	Y	Y			Y	Y		Y				Y	

[1] Odd SP not detected.

Consoles

Processor	03	T11	21	23		73		04	05/10	15/20	24	34	35/40	44	45/50/55	60	70	84	VAX
Module	**Various**	**DCT11**	**KXT11-A**	**KDF11-A**	**KDF11-B**	**KDJ11-A**	**KDJ11-B**	**KD11-D**	**KD11-B**	**KC11 KA11**	**KDF11-UA**	**KD11-E**	**KD11-A**	**KD11-Z**	**KD11-A KD11-D**	**KD11-K**	**KB11-B KB11-C**	**KDJ11-B**	**Various**
Programmer's Console:																			
Lights and switches	N			N	N		N	With KY11-LB	With KY11-J	Y	N	With KY11-LB	Y	N	Y	Y	Y	N	
ASCII console	Micro ODT	N	Y With Boot ROM	Micro ODT	Micro ODT	Micro ODT	Micro ODT	With M9312	N	N	Micro ODT	With M9312	N	MSD	N	N	With F.S. Contract	Micro ODT	
ASCII console memory addressing range in bits	16		16	18	18	22	22	18			18	18		22			22	22	
ASCII console could access GPRs	Y		Y	Y	Y	Y	Y	N			Y	N		Y			Y	Y	
Operator's Console:																			
Exists	Std			Std	Std		Std	With KY11-LA	Opt	With KY11-B	Std	With KY11-LA	Opt	Std	Opt	N	Opt	Std	
Remote Diagnosis:																			
Available from Field Service	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	Y	N	Y

Appendix B • Console ODT Command Languages

▪ Console ODT Command Set for PDP-11/84 and PDP-11/24

Descriptions of the PDP-11/84 and PDP-11/24 console terminal and a summarized table of the console ODT comands are found in Chapter 2 (PDP-11/84) and Chapter 4 (PDP-11/24). The ODT command language is identical for the PDP-11/84 and PDP-11/24 with the following two exceptions.

- The PDP-11/24 addresses 18 bits of physical memory. The PDP-11/84 addresses 22 bits of physical memory.
- The H command (ASCII 110) Toggle HALT flip-flop is only available with the PDP-11/24.

In order to describe the use of a command, other commands are mentioned before they have been defined. For the novice user, the following paragraphs should be skimmed first for familiarization and then reread for detail. The word "location" refers to a bus address, processor register, or the processor status word (PSW).

NOTE

In the following examples, the user's entry is in **boldface** (dark) type, while the response from the processor is not.

/(ASCII 057) Slash

This command (/) is used to open a bus address, processor register, or the processor status word and is normally preceded by other characters that specify a location. In response to /, console ODT prints the contents of the location (i.e., six characters) and then a space (ASCII 40). After printing is complete, console ODT waits either for new data for that location or for a valid close command. The space character is issued so that the location's contents and possible new contents entered by the user are legible on the terminal.

Example: **@001000/**012525<SPACE>

@	=console ODT prompt character.
001000	=octal location in the bus address space desired by the user (leading zeros are not required).
00100	=command to open and print contents of location.
12525	=contents of octal location 1000.
<SPACE>	=space character generated by console ODT.

The / command can be used without a location specifier to verify the data just entered into a previously opened location. The / is recognized only if it is entered immediately after a prompt character. A / issued immediately after the processor enters ODT mode causes a ?<CR><LF> to be printed because a location has not yet been opened.

Example: @**1000**/012525<SPACE>1234<**CR**><CR><LF>
@/001234<SPACE>

first line =new data of 1234 entered into location 1000 and location closed with <CR>.

second line =a / was entered without a location specifier and the previous location was opened to reveal that the new contents were correctly entered into memory.

<CR> (ASCII 015) Carriage Return

This command <CR> is used to close an open location. If a location's contents are to be changed, the user should precede the <CR> with the new data. If no change is desired, <CR> closes the location without altering its contents.

Example: @**R1**/004321<SPACE> <**CR**><CR><LF>
@

Processor register R1 was opened and no change was desired so the user issued <CR>. In response to the <CR>, console ODT printed <CR> <LF>@.

Example: @**R1**/004321<SPACE> **1234** <**CR**> <CR> <LF>
@

In this case the user desired to change R1, so new data, 1234, was entered before issuing the <CR>. Console ODT deposited the new data in the open location and then printed <CR><LF>@.

Console ODT echoes the <CR> entered by the user and then prints an additional <CR>, followed by a <LF>, and @.

<LF> (ASCII 012) Line Feed

This command <LF> is used to close an open location and then open the next contiguous location. Bus addresses and processor registers are incremented by 2 and 1 respectively. If the PSW is open when a <LF> is issued, the PSW is closed and a <CR><LF>@ is printed; no new location is opened. If the open location's contents are to be changed, the new data should precede the <LF>. If no data is entered, the location is closed without being altered.

Example: @**R2**/123456<SPACE> <**LF**><CR><LF>
R3/054321<SPACE>

In this case, the user entered <LF> with no data preceding it. In response, console ODT closed R2 and then opened R3. When a user has the last register, R7, open and issues <LF>, console ODT opens the beginning register, R0. When the user has the last bus address of a 32-Kword segment open and issues <LF>, console ODT opens the first location of that same segment. The user

who wishes to cross the 32-Kword boundary must reenter the address for the desired 32-Kword segment (i.e., console ODT is module 32 Kword). This operation is the same as that found on older PDP-11 consoles.

Example:
@**R7**/000000<SPACE> <**LF**><CR><LF>
R0/123456<SPACE>
or
@**577776**/000001<SPACE> <**LF**><CR><LF>
400000/125252<SPACE>

Unlike other commands, console ODT does not echo the <LF>. Instead it prints <CR>, then <LF> so the printing terminals operate properly. In order to make this easier to decode, console ODT does not echo ASCII 0, 2, or 10 either, but responds to these three characters with ?<CR><LF>@.

$ (ASCII 044) or R (ASCII 122) Internal Register Designator

Either character ($ or R) when followed by a register number, 0 to 7, or the PSW designator, S, will open that specific processor register.

The $ character is recognized to be compatible with ODT-11. The R character was introduced because it requires only one keystroke and because it is representative of what it does.

Example:
@**$0**/000123<SPACE>
or
@**R7/000123**<SPACE><**LF**>
R0/054321<SPACE>

If more than one character is typed (digit or S) after the R or $, console ODT uses the last character as the register designator. There is an exception, however, if the last three digits equal 077 or 477. ODT interprets this to mean the PSW rather than R7.

S (ASCII 123) Processor Status Word (PSW)

This designator is for opening the PSW and must be employed after the user has entered an R or $ register designator.

Example:
@**RS**/100377<SPACE>**0**<**CR**><CR><LF>
@/000020<SPACE>

Note the trace bit (bit 4) of the PSW cannot be modified by the user. This is done so that PDP-11 program debug utilities (e.g., ODT-11) that use the T bit for single-stepping are not accidentally harmed by the user.

If the user issues a <LF> while the PSW is open, the PSW is closed and ODT prints a <CR><LF>@. No new location is opened in this case.

G (ASCII 107) Go

This command (G) is used to start program execution at a location entered immediately before the G. This function is equivalent to the LOAD ADDRESS and START switch sequence on older PDP-11 consoles.

Example: **@200G**<NULL><NULL>

The console ODT sequence for a G, after echoing the command character follows.

1. Print two nulls (ASCII 0) so that the bus initialization that follows does not flush the G character from the double-buffered UART chip in the serial line interface.
2. Load R7 (PC) with the entered data. If no data is entered, 0 is used. R7 is set equal to 200 and that is where program execution begins.
3. The PSW and floating-point status register contained in the MMU are cleared to 0.
4. The bus is initialized.
5. The service state is entered by the processor. If there is anything to be serviced, it is processed. If the HALT signal is asserted, the processor reenters the console ODT state. This feature is used to initialize a system without starting a program (R7 is altered). A user who wants to single-step a program, can execute it by issuing a G and then successive P commands, all done with the HALT signal asserted (either by the HALT switch or via the H command).

P (ASCII 120) Proceed

This command (P) is used to resume execution of a program and corresponds to the CONTINUE switch on older PDP-11 consoles. No programmer-visible machine state is altered using this command.

Example: **@P**

The PDP-11 processor is started immediately after the transmission of the P to the terminal console has begun. If a RESET instruction is executed while the P is transmitting, the echo of the P may be lost.

Program execution resumes at the address pointed to by R7. After the P is echoed, the console ODT state is left and the processor immediately fetches the next instruction. If the HALT signal is asserted, it is recognized at the end of the instruction (during the service state) and the processor enters the console ODT state. Upon entry, the content of the PC (R7) is printed. In this fashion, a user can single-step an instruction through a program and get a PC trace displayed on the terminal.

Control-S (ASCII 023) Binary Dump

This command (Control-S) is used for manufacturing test purposes and is not a normal user command. It is described here to explain the machine's response if the command is accidentally invoked. Control-S is intended to more efficiently display a portion of memory than the / and <LF> commands can display it. The protocol follows.

1. After a prompt character, console ODT receives a control-S command and echoes it.
2. The host system at the other end of the serial line must send two 8-bit bytes that console ODT interprets as a 16-bit starting address. These two bytes are not echoed.

   The first byte specifies starting address <15-08> and the second byte specifies starting address <07-00>. Bus address bits <17-16> are always forced to be 0; the dump command is restricted to the first 32 Kwords of address space.
3. After the second address byte has been received, console ODT outputs 10 bytes to the serial line starting at the address previously specified. When the output is finished, console ODT prints <CR><LF>@.

If a user accidentally enters this command, it is recommended, in order to exit from the command, that the user resets the terminal and enters two @ characters (ASCII 100) as a starting address. After the binary dump, an @ prompt character is printed.

H (ASCII 110) Toggle HALT Flip-Flop (PDP-11/24 only)

Programs are often debugged by single-instruction stepping them. In the PDP-11/24, this may be accomplished by setting the HALT/BOOT switch to the HALT position and using the P command to single-instruction-cycle the PDP-11/24. The same result as setting the HALT switch may be realized by typing H. An internal flip-flop is set, simulating the action of the HALT switch. P will now single-cycle the PDP-11/24. After the debugging execution is completed, H must be typed once more. This will clear the internal flip-flop, and the next P will cause the PDP-11/24 to run at full speed.

▪ ASCII Console for the PDP-11/44

The PDP-11/44 serial console is a standard feature that replaces the "lights and switches" programmer's console of earlier processors with logic that interprets ASCII characters to perform equivalent panel functions.

Physically, the I/O port used for the serial console function is shared with the standard system terminal (also called the system console), and is mode (or state) switchable by typing ASCII characters on the system terminal (the LA120 or equivalent that serves as the system console/programmer console).

In this section, "console state" defines the serial console mode of operation in which ASCII commands are interpreted and result in the programmer's console functions (deposit, examine, halt, continue, etc.) being performed. The term "program I/O state" will be used to refer to that state in which the LA120 functions as the standard system terminal, or the system console.

NOTE

The console state can be entered only when the key switch is not in the local disable position.

Console State

The console state is entered by typing a reserved input character, Control P (ASCII <CTRL>P <020> or <220>). This is also called the console break character. The console state is also entered when the CPU halts. It can be entered only when the front panel key switch is in the local position. The reserved character is not passed to a running program, and console state is entered after printing the current output character, if any. While in the console state, all input characters are interpreted by the console logic as commands to the CPU control interface. The console performs all character echoing while in the console state.

A program running in the processor is inhibited by the console logic from sending or receiving any characters. This is accomplished by inhibiting the "ready" and "done" bits from being set. The console state is exited to the program I/O state by typing a specific console command such as CONTINUE, START, or BOOT. If there is no console command in execution, a front panel CONTINUE will cause exit from the console state. Turning the front panel keyswitch to the local disable position will cause exit. The beginning of a powerdown sequence will also cause exit from the console state.

NOTE

When in console state, if a program just sends output to the printer without testing status bits, the characters will be printed if the logic happens to be ready.

Program I/O State

The program I/O state is entered from the console state by typing the CONTINUE command. A running program will then resume any input/output that might have been interrupted by the console break character. Any ASCII character may be output by the program, and any ASCII character, except the console break character, may be input to the program. Character echoing is the responsibility of the CPU software in program I/O state.

The program I/O state is exited to the console state by typing the console break character, or by CPU execution of a HALT.

▪ Console Command Language for the PDP-11/44

This section lists the control characters and special characters recognized by the console and describes their functions.

ADDRESS

The ADDRESS command is used to redisplay the results of a previous console operation. The command provides a convenient method of calculating the effective address of modes 67 and 77 offset addresses.

Syntax: >>>A[ddress]

Example:
```
>>>A
    P 00-001000 000240
    As an index 001242
>>>
```

The first line displays the last physical address used by the console as noted by the P. If an internal register was addressed, this would be an M. The data is the last word examined or deposited by the console.

The second line contains the address that would be referenced if the current word was used as the index for a mode 67 instruction.

Example:
```
>>>E 1000/N:3
      P  00001000 012767
      P  00001002 000301
      P  00001004 176560
>>>A
      P  00001004 176560
    As an index      177566
```

The MOVE instruction at addresses 1000, 1002, and 1004 has a source mode of 27 (immediate) and a destination mode of 67 (offset from the PC).

BOOT

The BOOT command allows the user to boot from a system device other than the default system device. This command is executed only if the CPU is halted. Otherwise, an error message is generated. Console state is exited before boot execution is continued.

Syntax: >>>B[oot][<SP><DEVICE-NAME>][:][<Qualifier>]

If no device name is given, the console will perform the boot sequence for the default system device. This is the equivalent of using the front panel BOOT switch.

Example: >>>B DT2:/NO

BREAK

The BREAK command, when supplied with a physical address, sets a hardware breakpoint. If this breakpoint is set and the CPU or an I/O device issues a read or write to the specified address, the CPU will halt. If no address is supplied, the breakpoint will be cleared. The breakpoint remains in effect until it is explicitly cleared or the system is powered off.

Syntax: >>>BR[eak] [<address>] [<Qualifier>]

The optional address qualifier may be used to specify virtual address space and the address translation will be made once using the current mapping. If the mapping is subsequently changed, the breakpoint will remain at the old physical address.

The cache address-match registers are used by this command. Therefore, the cache module (M7097) must be installed in the system. These registers (17777750/17777752) should not be explicitly addressed while the breakpoint is in effect nor should a breakpoint be attempted using these addresses.

Example: >>>BR1000/KI
(Console) Break at 001004
>>>

The CPU is halted and the PC is displayed when the address specified is encountered. It may be the PC of the first or second instruction following the breakpoint. The address may be any word in the instruction stream, an explicit operand address, or an implicit stack access. If the address was accessed by DMA activity, the contents of the PC may be entirely unrelated.

CONTINUE

When the CONTINUE command is issued, the CPU begins instruction execution at the address currently contained in the CPU program counter (PC) or continues execution if already running. CPU initialization is not performed. Additionally, the console enters program I/O state at the same time as issuing the CONTINUE to the CPU.

Syntax: >>>C[ontinue]

This command may be used to return the console to program I/O state even if the CPU was already running.

DEPOSIT

The DEPOSIT command is used to modify the contents of memory, the general purpose registers, PDP-11 programmer accessible registers, the PDP-11 micro PC, and various I/O registers. Also, it will deposit data into the address specified. The address space will depend upon the qualifiers specified with the command.

Syntax: >>>D[eposit] [<address><sp>] <data> [<Qualifier>]

An address may be specified with the DEPOSIT command; if no address is specified, the address defaults to <+> (preincrement).

Initiating deposits while the processor is running is illegal unless the deposit is to the console switch register (D<sp>SW<data><cr>).

```
Example:  >>>D 1000 240
          >>>D 240
          >>>
```

The first DEPOSIT command loads 000240 into location 0001000. The second DEPOSIT loads 000240 into location 0001002.

EXAMINE

The EXAMINE command allows the user to display the contents of memory, the general purpose registers, PDP-11 programmer accessible registers, the PDP-11 internal registers, and various I/O registers.

Syntax: >>>E[xamine] [<address>]

The EXAMINE command can be used without the argument. In this case, the address defaults to <+> (preincrement).

Examines are legal while the processor is running. The console will respond by printing the eight-digit physical address examined followed by the six-digit octal data contained in that location. Upon completion of the examine, the console will respond with the console prompt (>>>).

```
Examples:  >>>E 1000          Examine location 1000.
           00001000 002625
           >>>E               Examine the next location.
           00001002 005646
           >>>E PC            Examine the PC.
           >>>G
           00000007 001000
```

FILL

Syntax: >>>F[ill] count

The FILL command must be followed by a numeric count between 0 and 377. The count is equal to the number of null characters to be echoed following a <CR>.

```
Example:  >>>F 100
          >>>F 0
          >>>
```

The first command sets the fill count to 100 (64 decimal). The second command sets the fill count back to 0.

On powerup, the fill count is set to 0. Upon completion of the FILL command, the console responds with the console prompt (>>>).

HALT

The HALT command is used to halt the processor at the completion of its current instruction. After the processor has halted, the PC is displayed.

Syntax: >>>H[alt]
Example: >>>H
Halted at 001000
>>>

HELP

This command provides a listing of all the syntax recognized by the console. The binary load command is not listed.

Syntax: >>>HE<lp>

INITIALIZE

The INITIALIZE command causes the entire PDP-11 bus and the I/O to be reset to the conditions present after initial powerup.

Syntax: >>>I[nitialize]

MICROSTEP

The MICROSTEP command causes the PDP-11 processor to execute one instruction in the microprogram. The updated PC will be printed.

Syntax: >>>M[icrostep] [count] [qualifier]

If either a count or a qualifier is supplied, the PDP-11 will take that many steps. The count is decremented after each microinstruction is performed. When the count equals 0, the console will print the last micro PC and the console prompt (>>>). The console is then in spacebar step mode and an additional microinstruction is performed each time the spacebar or tab key is pressed.

Example: >>>M 2
micro PC = 000015
micro PC = 000263

NEXT

The NEXT command causes the PDP-11 processor to execute one instruction in the program. The updated PC will be printed.

Syntax: N[ext] [count]

If used without an argument or qualifier, the NEXT command will execute one instruction. If either a count or a qualifier is supplied, the PDP-11 will execute

that many instructions. Immediately after the NEXT command has been executed, additional instructions may be executed simply by pressing the spacebar or tab key.

NOTE

Interrupts are disabled while single stepping the PDP-11.

Example:
```
>>>N
    PC=001002
>>>N = 2
    PC=001004
    PC=001010
>>>
```

REPEAT

The REPEAT command will repeatedly execute <command>.

Syntax: >>>R[epeat] <comand>

R[epeat] will repeatedly execute <command> until it is terminated by CTRL-C.

START

The START command causes the PDP-11 to begin executing its program at the address specified. The PDP-11 is initialized, the new PC deposited, the breakpoint (if any) is set, and the PDP-11 continued.

Syntax: S[tart] address

An address must be supplied with the START command.

Example:
```
>>>S 1000
    (program)
```

TEST

The TEST command causes the console to perform its self test. By using the /E qualifier, the test will include console interaction with the PDP-11.

Syntax: >>>T[est] [/Extensive>]

Example:
```
>>>T
>>>
```

Errors during this phase of testing produce no error messages. Success of this test is indicated by the console prompt.

```
>>>T/E
    Halted at 000000
>>>
```

Errors during the /E[xtensive] part of the test are indicated by the printing of an error message. If no errors occur "Halted at 000000" will be printed at the end of each pass.

NOTE

/E[xtensive] version of the TEST destroys the contents of the switch register and PDP-11 memory. Unless specially designed, the PDP-11 program is not restartable after T[est]/E[xtensive].

BINARY LOAD

The BINARY LOAD command instructs the console to prepare to load or unload <count> binary data bytes starting from location <address>. Only an even byte <count> may be used.

Syntex: >>>X<address><sp><count><cr>
Example: >>>X 1000 866

A <count> without bit 18 set (0 XXX XXX) will cause a BINARY LOAD of the count field. A <count> with bit 18 set (1 XXX XXX) will cause a BINARY UNLOAD. In this case, the remaining bits in the count field are considered an unsigned positive number indicating the number of bytes to unload.

Command Qualifiers

QUALIFIER	COMMAND APPLICABILITY	DESCRIPTION
/IR	D[eposit] E[xamine] M[icrostep] N[ext] T[est]	Repeats the command continuously. Halts on error or CTRL-C.
/NO[diagnostic]	B[oot]	Disables running diagnostics only if a device specifier is used following the command.
/SC[ope]	DEPOSIT EXAMINE HALT TEST	Retry the command continuously (will not halt on most errors) CTRL/C will suspend output.
/E[xtensive]	TEST	Test CPU-console interaction.
/N:<count>	DEPOSIT EXAMINE MICROSTEP TEST	Repeat the command for <count> iterations.
/CB	EXAMINE	Cache bypass
/TB	DEPOSIT EXAMINE INITIALIZE	Take-bus

Addressing Qualifiers

To access virtual memory these qualifiers may be appended to <address>.

Syntax Example: >>>E 1000/K/N:2
KI 00001000 000001
KI 00001002 000776

QUALIFIER	DESCRIPTION
/P	Physical bus addresses (memory or I/O).
/K[i]	Kernel instruction space.
/KD	Kernel data space.
/SD	Supervisor data space.
/S[i]	Supervisor instruction space.
/U[i]	User instruction space.
/UD	User data space.
/V[i]	Use the instruction space specified by the current mode bits of the PSW.
/VD	Use the data space specified by the current mode bits of the PSW.

<address> Arguments

The <address> in D[eposit] and E[xamine] commands can be any of the following arguments.

ARGUMENTS	DESCRIPTION
0–7	1 to 8 octal digits.
—	Autodecrement.
*	Use the last address again.
+	Autoincrement (default).
@	The last data examined becomes the address.
/G[n]	General purpose registers (n=0–17 octal).
/M[n]	Machine dependent registers (n=0–11 octal)* The only machine-dependent register that can be deposited into is the CPU micro PC register (address 00000004). * 0–10 if the M7096 module is below revision C.

SW[r]	Switch register. Equal to /P 17777570.
PC	Program counter. Equal to /G 7.
PS[w]	Processor status word. Equal to /P 17777776.
KS[p]	Kernel stack pointer. Equal to /G 6.
SS[p]	Supervisor stack pointer. Equal to /G 16.
US[p]	User stack pointer. Equal to /G 00000017.

Execution Errors

?Halt CP	The command cannot be executed when the CPU is running.
?No ROM for that	The CPU does not contain a bootstrap ROM for the specified device.
?CP did not start	A hardware failure prevented the CPU from performing the requested command.
?UnBREAKable	It is illegal to set the breakpoint to physical address 0, any general purpose register, or machine-dependent register.
?Bus timeout error	The address you are attempting to breakpoint is nonexistent. There is no cache memory in the CPU. The specified address is not responding.
?Parity error	Parity is enabled and the address you are attempting to breakpoint or examine contains bad parity.
?μStep CP	Breakpoints cannot be set while the CPU is at microaddress 015.
?Too big	Attempting to deposit, examine or breakpoint based on a virtual address larger than 177777. Deposit/examine a general-purpose register greater than 17 or machine dependent register greater than 11 (10 for V3.XXB machines). The requested fill count was greater than 377 (225 decimal).
?Access aborted	Attempted to breakpoint/deposit/examine a virtual address that is not mapped.
%Wrote RO page	Deposit to a virtual page that is read-only. The deposit is performed. This is a warning message.

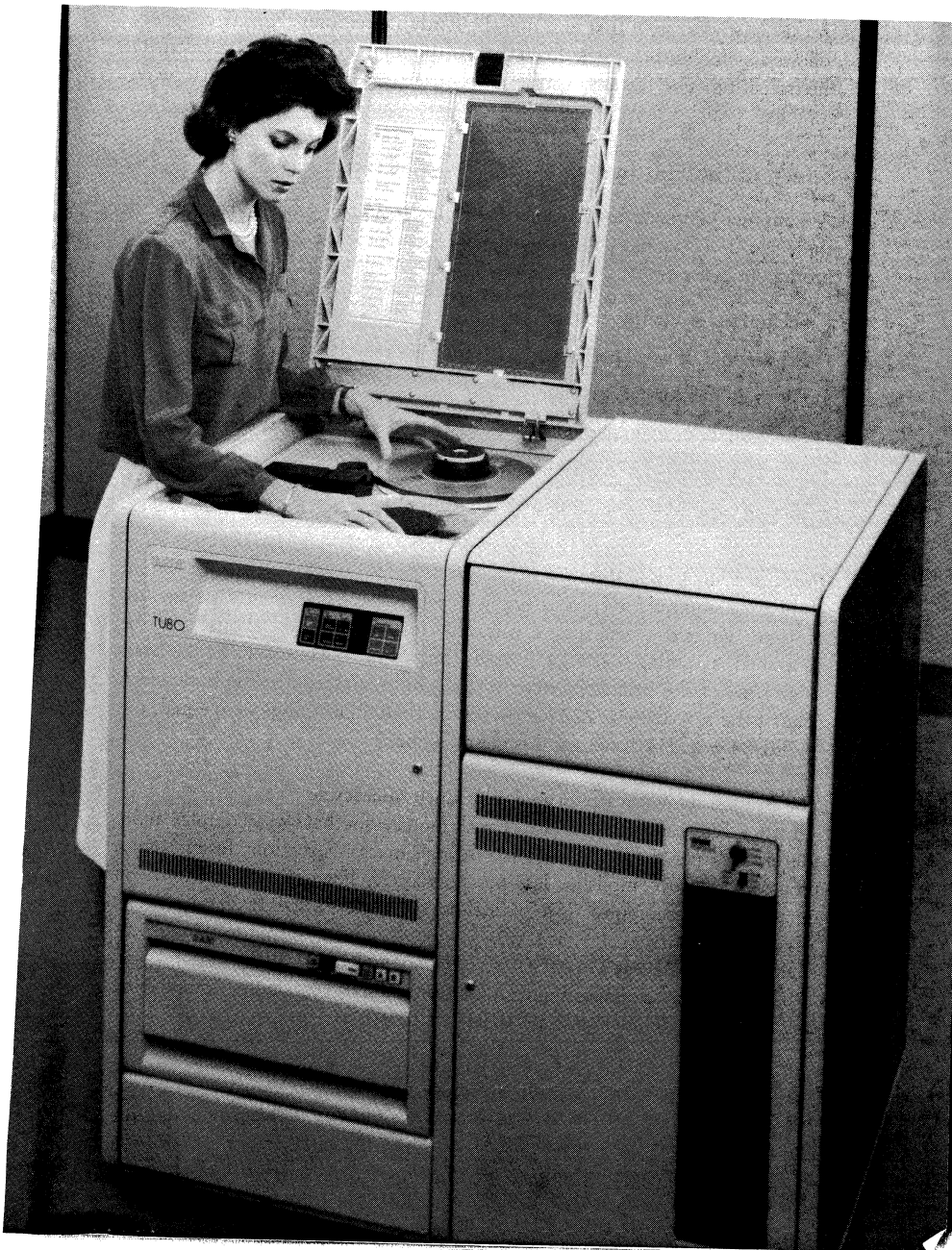
?Read-only	Only the micro PC register may be deposited. All other machine dependent registers are read-only.
?Failed to halt	The console attempted to halt the CPU but could not do so after 500 microseconds.
?Already halted	The CPU is already halted.
?Rn Err	The console is not certain of the RUN/HALT state of the CPU. The CPU did not halt after 500 microseconds following an EXAMINE command.
?PX Err	The pax address/data lines appear to be stuck high or low.

Notes

1. The message "Console V3.40x" will be printed on powerup where x will be the revision level of the M7096 module installed in the system.
2. Entering a 22-bit address is not required. Addresses will be expanded to 22-bits (e.g., 1000+00001000).
3. Entering a 16-bit data word is not required. Data will be expanded to 16-bits (e.g., 240=000240).
4. Spaces are required to separate numeric groups and alpha character groups to avoid ambiguity.

Examples:	Right	>>>D PSW340/IR
	Wrong	>>>DPSW340/IR
	Right	>>>D1000 12737/TB
	Wrong	>>>D100012737/TB

Appendix C • Instruction Timing



It is often useful to know the amount of time required to execute a particular instruction or series of instructions. The calculation of this time is straightforward but dependent on a variety of factors. These factors break down into two main categories—speed of the hardware and complexity of the instruction.

In this appendix, we'll first examine the hardware features that affect speed, followed by tables that break down the instructions and allow you to calculate the execution time for any instruction as executed by the various hardware systems.

▪ Speed of the Hardware

A computer system is built of many separate hardware subsystems. Each of these can affect the rate at which instructions are processed. Some of the most prominent factors are

- The processor microcycle rate.
- The amount of work performed with each microcycle.
- The main memory access time.
- The number of main memory cycles that can be avoided entirely.
- Whether there is DMA activity on the system bus.
- Timing variations peculiar to a given system.

Processor Microcycle Rate

If the time required for a processor microcycle is shortened, clearly the amount of time required to execute a complete macroinstruction should also be shortened, assuming nothing else stalls the processor. For example, the basic clock time of the PDP-11/24 microcycle is 300 nanoseconds; the basic clock time of the PDP-11/84 is 222 nanoseconds. If all other things were equal, the PDP-11/84 would be faster for this reason alone.

The Amount of Work Performed with Each Microcycle

All other things are not necessarily equal. A processor that can accomplish more useful work per microcyle will also execute macroinstructions faster than a processor that accomplishes less per microcycle. For example, a PDP-11/84 microcycle can accomplish up to three times the work as a PDP-11/24 microcycle.

Main Memory Access Time

While the processor is waiting for data from memory, it is not accomplishing useful work. Faster memories mean less waiting time.

Minimizing the Number of Main Memory Cycles

The inclusion of fast buffer memories between the processor and main memory can mean that fewer references need to go all the way to main memory for data. This speeds execution while freeing up main memory cycles for use by DMA devices. The PDP-11/84 uses two methods to minimize the number of main memory cycles—cache memory and the instruction buffer.

Cache Memory

Both the PDP-11/84 and the -11/44 contain a buffer memory between the processor and the main memory. This buffer memory attempts to store the data and instruction words most frequently required by the processor. This buffer is referred to as the *cache memory*. Any processor data request satisfied by the cache memory takes much less time than a request that can be satisfied only from main memory. When the requested data is found in the cache, the operation is referred to as a *cache hit*. When the requested data is not found in the cache, the operation is referred to as a *cache miss*, and the request automatically passes to main memory.

In addition, every time the PDP-11/84 must go to main memory, two words will be returned to the cache. This is useful because there is a high probability that the second word will be used soon.

The PDP-11/24 does not contain a cache memory. All memory requests result in a read or write of main memory.

The Instruction Buffer

In addition to the speed improvement provided by its cache, the PDP-11/84 attempts to predict the address of the next instruction word required by the processor. Because the processor is normally accessing instruction words in ascending order (as directed by the Program Counter), this prediction is generally successful. Separate hardware within the processor performs this prediction and uses "spare" memory or cache cycles to access the next instruction word. This next instruction word is then stored in the *instruction buffer*. Should the prediction prove true, the word is already waiting and the processor need not stop to fetch that next word. If, however, the program branches, the prediction will prove false and the processor will need to stop while the real next instruction word is fetched from memory.

Any read access using the PC is referred to as an Instruction-stream (I-stream) read. Read accesses that do not use the PC cannot be predicted in this fashion, and are referred to as data reads.

Neither the PDP-11/24 nor -11/44 contain an instruction buffer, so all read operations, whether I-stream or not, act like data reads.

Effective Memory Access Time

The table below indicates the time to access data from each of the different data sources (e.g., the instruction buffer, the cache memory, or main memory), for each of the different processors (i.e., the PDP-11/84, -11/44, and -11/24).

	I-stream reads	Data reads	Data writes
11/84, data in...			
Instruction buffer	0 ns		
Cache memory	222 ns	222 ns	
Main memory	1000 ns	1000 ns	680 ns
11/44, data in...			
Cache memory	90 ns	90 ns	
Main memory	450 ns	450 ns	150 ns
11/24, data in...			
Main memory	450 ns	450 ns	150 ns

Direct Memory Access (DMA)

If the processor must read or write a main memory word, and the UNIBUS is already busy servicing a direct memory access (DMA), then the processor must wait until the DMA completes. This can greatly extend the processor's effective memory access time.

In most circumstances, the Dual Tag Store of the PDP-11/84 allows the processor to continue to read from the cache memory unimpeded by DMA activity. The processor must wait only if a cache miss or a main memory write occurs or the DMA attempts to alter data already stored in the processor's cache memory.

The PDP-11/44 and -11/24 will be stalled any time a memory read or write is required and a DMA is occurring on the UNIBUS.

Timing Variations in a Given System

Finally, the speed of the various hardware systems varies from unit to unit. The clock of the PDP-11/84 processor is timed by a quartz crystal and so is very accurate, but the memory system is not so precise. The clocks of the PDP-11/24 and -11/44 processors may vary as much as ±5 percent from sample to sample.

▪ Instruction Complexity

PDP-11 processors fetch instructions from main memory. These instructions can change the "state" of the CPU, or manipulate data stored in main memory. The instructions vary in complexity and, therefore, in execution speed. Let's examine the execution of a few PDP-11 instructions.

Increment (INC)

This instruction causes the operand to be replaced by the operand plus one, that is, the operand is incremented by one. The operand may be a CPU general register or a byte or word of main memory. The first step in the execution of any instruction is to fetch the opcode from main memory. Contained within the opcode is the field specifying the mode to be used in addressing the operand. Also specified is the general register that will combine with the addressing mode to tell the CPU where the operand will be found.

If the operand is contained within a general register (address mode 0), then the register can be quickly incremented, and the INC instruction is complete.

If any other addressing mode was selected, the operand is contained in main memory. First, the address in memory of the operand must be determined. This requires zero, one, or two reads of main memory, depending on the particular addressing mode. Then the operand can be read from that address. The data is incremented within the CPU and the updated data written back into main memory. The INC instruction always returns the incremented data to the same location as previously read, so the address calculation need not be repeated.

Move (MOV)

The MOV instruction is used to copy data from one location to another. The location that supplies the data is called the *source*. The location to receive the data is called the *destination*.

As in all instructions, the opcode is first fetched from memory. This time, the opcode contains two addressing fields. One specifies the addressing mode and register for the source operand, and the other specifies the addressing mode and register for the destination operand.

The exact order of operations varies from one member of the PDP-11 family to another but, in general, the following operations take place:

The source field is evaluated to supply the address of the source operand (i.e., where the MOV instruction will find the data). Remember that this requires zero, one, or two memory reads, depending on the addressing mode. The destination field is then evaluated to determine the address of the destination operand (i.e., where the MOV instruction will put the copy of the data). The source operand itself is then read, and the destination operand written. If the source operand is in main memory, this requires reading main memory. If the destination operand is in main memory, this requires writing main memory.

Add

This is another two-operand instruction. It differs from the MOV instruction only in that the destination operand is replaced with the sum of the source and destination operands. In other words, both the source and destination operands must be read, then the two data words added together, and the result

rewritten to the destination operand. Thus, if the destination operand is in main memory, ADD requires one more read of main memory than an equivalent MOV instruction.

Emulator Trap (EMT)

This instruction does not require the specification of any addressing modes. This is because all of the operands are implicitly specified by the opcode itself.

The instruction allows the software to perform a trap (similar to a hardware interrupt). This instruction is generally used to call monitor or kernel routines.

As always, the opcode is first fetched from memory. Then the old program counter (PC) and processor status (PS) are pushed onto the SP stack. The PC and the PS are then loaded from the EMT trap vector (at location 000 030). (This is a simplified explanation, not taking into account memory management.)

Beyond the opcode fetch, EMT therefore requires two main memory writes as well as two main memory reads to execute completely.

General Method to Calculate Instruction Timing

In general, all instructions executed by the PDP-11 require an opcode fetch, and most require that some data be manipulated. The data manipulation consists of one or more of source operand access, destination operand access, destination operand write-back, and miscellaneous implied reads and writes.

The total execution time of an instruction can be found by summing up the time required for each of the basic operations listed above. The tables that follow provide the specific timings for each of these basic operations.

The three-operand CIS instructions are different and are not covered in this appendix.

Overlap of Phases

Certain simple addressing modes allow much work to be accomplished in a single microcycle. For example, most PDP-11 processors can execute the instruction "ADD R1, R0" in a single microcycle. In that single microcycle, R1 and R0 are presented to the adder and the result returned to R0. Thus, the source, destination, execution, and writeback all take place concurrently.

Addressing Modes

The various addressing modes require various amounts of work. Mode 0 (operand in general register) requires essentially no work and is the fastest addressing mode. Mode 7 (indexed, deferred) is the most complex addressing mode, requiring the most work and execution time. Use of the various addressing modes is described in the *PDP-11 Architecture Handbook*. The tables below document the number of I-stream reads, data reads, and data writes performed by each of the source and destination addressing modes as

well as their contribution (in microseconds) to the overall instruction execution time. Separate tables document

- Source addressing modes.
- Destination modes for BIT(B), CMP(B), TST(B).
- Destination modes for JMP.
- Destination modes for JSR.
- Destination modes for MOV(B) and CLR(B).
- Destination modes for all other instructions.

Source Addressing Modes

The source operand is never written, so a source operand access simply consists of read operations.

Source Mode	Macro-11 syntax	I-stream reads	Data reads	Data writes	CPU activities required	11/84	11/44	11/24
00-07	R	0	0	0	<None>	0	.18*	0
10-17	@R or (R)	0	1	0	Access operand	.44	.18	.60
20-26	(R)+	0	1	0	Access operand Increment R	.44	.36	.60
27	(PC)+ or #n	1	0	0	Access operand, Increment PC	.22	.36	.60
30-36	@(R)+	0	2	0	Access addr of operand Access operand Increment R	.89	.54	1.20
37	@(PC)+ or @#n	1	1	0	Access addr of operand Access operand Increment PC	.67	.54	1.20
40-46	-(R)	0	1	0	Decrement R Access operand	.67	.36	3.90
47	-(PC)	0	1	0	Decrement PC Access operand	1.33	.36	3.90
50-56	@-(R)	0	2	0	Decrement R Access addr of operand Access operand	1.11	.54	1.50
57	@-(PC)	0	2	0	Decrement PC Access addr of operand Access operand	1.78	.54	1.50
60-67	INDEX(R)	1	1	0	Access index word Sum with R Access operand	.89	.72	1.50
70-77	@INDEX(R)	1	2	0	Access index word Sum with R Access addr of operand Access operand	1.33	.90	2.10

*0 if destination mode also equals 00-07.

Destination Addressing Modes for BIT(B), CMP(B), and TST(B) (read-only) These instructions never modify the destination operand. Therefore most PDP-11's never write the destination operand. Other older models write back the exact same data as read. See the PDP-11 difference list (Appendix A) for details.

Destination Mode	Macro-11 syntax	I-stream reads	Data reads	Data writes	CPU activities required	11/84 Single Op	11/44	11/24	11/84 Double Op
00-07	R	0	0	0	<None>	0	.18*	0	0
10-17	@R or (R)	0	1	0	Access operand	.44	.18	.90	.67
20-26	(R)+	0	1	0	Access operand Increment R	.44	.36	.90	.67
27	(PC)+ or #n	1	0	0	Access operand, Increment PC	.22	.36	.90	.44
30-36	@(R)+	0	2	0	Access addr of operand Access operand Increment R	.89	.54	1.20	1.11
37	@(PC)+ or @#n	1	1	0	Access addr of operand Access operand Increment PC	.67	.54	1.20	.89
40-46	-(R)	0	1	0	Decrement R Access operand	.67	.36	.90	.89
47	-(PC)	0	1	0	Decrement PC Access operand	1.55	.36	.90	1.78
50-56	@-(R)	0	2	0	Decrement R Access addr of operand Access operand	1.11	.54	1.50	1.33
57	@-(PC)	0	2	0	Decrement PC Access addr of operand Access operand	2.00	.54	1.50	2.22
60-67	INDEX(R)	1	1	0	Access index word Sum with R Access operand	.89	.72	1.50	1.11
70-77	@INDEX(R)	1	2	0	Access index word Sum with R Access addr of operand Access operand	1.33	.90	2.10	1.55

*0 if source mode also equals 00-07.

Destination Addressing Modes for JMP

The JMP instruction performs one less read than an equivalent MOV to PC instruction, and is implemented with its own microroutines.

Destination Mode	Macro-11 syntax	I-stream reads	Data reads	Data writes	CPU activities required	11/84 Single Op	11/44	11/24
00-17	R	N/A	N/A	N/A	Illegal instruction—Trap through 10	N/A	N/A	N/A
10-11	@R or (R)	0	0*	0	PC←R	.89	.54	1.50
20-26	(R)+	0	0*	0	PC←R Increment R	1.33	.90	1.80
27	#n	0	0*	0	PC←PC Increment R	1.33	.90	1.80
30-36	@(R)+	0	1*	0	Access operand PC←operand Increment R	1.11	.90	1.80
37	@#n	1	0*	0	Access operand PC←operand	1.11	.90	1.80
40-47	-(R)	0	0*	0	Decrement R PC←R	1.11	.72	1.80
50-57	@-(R)	0	1*	0	Decrement R Access operand PC←operand	1.33	.90	2.20
60-66	Index(R)	1	0*	0	Access index word PC←R + index	1.33	.90	2.20
67	Symbol	1	0*	0	Access symbol word PC←PC + symbol + 2	1.11	.90	2.1
70-76	@Index(R)	1	1*	0	Access index word $Addr_t$←R + index Access T PC←T	1.55	1.26	2.7
77	@ Symbol	1	1*	0	Access symbol word $Addr_t$←PC + symbol + 2 Access T PC←T	1.55	1.26	2.7

*PDP-11/84 performs one additional read (e.g., 1 instead of 0)

Destination Addressing Modes for JSR

The Jump-to-Subroutine instruction is very similar to the JMP instruction, except that JSR must push the old contents of the destination register onto the SP stack. This requires extra microcycles and one data write. These extra activities are not listed under the "CPU activities" column.

Destination Mode	Macro-11 syntax	I-stream reads	Data reads	Data writes	CPU activities required	11/84 Single Op	11/44	11/24
00-17	R	N/A	N/A	N/A	Illegal instruction—Trap through 10	N/A	N/A	N/A
10-11	@R or (R)	0	0*	1	PC←R	2.00	1.26	2.70
20-26	(R)+	0	0*	1	PC←R Increment R	2.22	1.44	3.00
27	#n	0	0*	1	PC←PC Increment R	2.22	1.44	3.00
30-36	@(R)+	0	1*	1	Access operand PC←operand Increment R	2.22	1.44	3.00
37	@#n	1	0*	1	Access operand PC←operand	2.00	1.44	3.00
40-47	-(R)	0	0*	1	Decrement R PC←R	2.22	1.26	3.00
50-57	@-(R)	0	1*	1	Decrement R Access operand PC←operand	2.22	1.44	3.3
60-66	Index(R)	1	0*	1	Access index word PC←R + index	2.22	1.62	3.3
67	Symbol	1	0*	1	Access symbol word PC←PC + symbol + 2	2.00	1.62	3.3
70-76	@Index(R)	1	1*	1	Access index word $Addr_T$←R + index Access T PC←T	2.66	1.80	3.90
77	@ Symbol	1	1*	1	Access symbol word $Addr_T$←PC + symbol + 2 Access T PC←T	2.66	1.80	3.90

*PDP-11/84 performs one additional read (e.g., 1 instead of 0)

Destination Addressing Modes for MOV(B) and CLR(B) (write-only)

The Move and Clear instructions never need to access the destination data, because that data is merely overwritten. Therefore, some PDP11 processors suppress the reading of the destination data for MOV(B), CLR(B), or both. This speeds execution. Other processors may read the data and then discard it. See the PDP-11 difference list (Appendix A) for details.

Destination Mode	Macro-11 syntax	I-stream reads	Data reads	Data writes	CPU activities required	11/84	11/44	11/24
00-07	R	0	0	0	<None>	1.11	.18*	0
10-16	@R or (R)	0	0	1	Write operand	.44	.36	1.20
17	@PC or (PC)	0	0	1	Write operand	1.33	.36	1.20
20-26	(R)+	0	0	1	Write operand Increment R	.44	.54	1.20
27	(PC)+ or #n	0	0	1	Write operand, Increment PC	1.33	.54	1.20
30-36	@(R)+	0	1	1	Access addr of operand Write operand Increment R	.89	.72	1.50
37	@(PC)+ or @#n	1	0	1	Access addr of operand Write operand Increment PC	.67	.72	1.50
40-46	-(R)	0	0	1	Decrement R Write operand	.67	.54	1.20
47	-(PC)	0	0	1	Decrement PC Write operand	1.55	.54	1.20
50-56	@-(R)	0	1	1	Decrement R Access addr of operand Write operand	1.11	.72	1.80
57	@-(PC)	0	1	1	Decrement PC Access addr of operand Write operand	2.00	.72	1.80
60-67	INDEX(R)	1	0	1	Access index word Sum with R Access operand	.89	.90	1.80
70-77	@INDEX(R)	1	1	1	Access index word Sum with R Access addr of operand Access operand	1.33	1.08	2.40

*0 if source mode also equals 00-07.

Destination Addressing Modes for All Other Instructions (read-write)
All other instructions read the old destination operand, modify the data, and then write back new data.

Destination Mode	Macro-11 syntax	I-stream reads	Data reads	Data writes	CPU activities required	11/84	11/44	11/24
00-06	R	0	0	0	<None>	0	.18*	0
07	R	0	0	0	<None>	1.11	.18*	0
10-16	@R or (R)	0	1	1	Access operand Write-back operand	.67	.36	1.50
17	@PC or (PC)	0	1	1	Access operand Write-back operand	1.55	.36	1.50
20-26	(R)+	0	1	1	Access operand Increment R Write-back operand	.67	.54	1.50
27	(PC)+ or #n	0	1	1	Access operand, Write-back operand Increment PC	1.55	.54	1.50
30-36	@(R)+	0	2	1	Access addr of operand Access operand Write back operand Increment R	1.11	.72	1.80
37	@(PC)+ or @#n	1	1	1	Access addr of operand Access operand Write-back operand Increment PC	.89	.72	1.80
40-46	-(R)	0	1	1	Decrement R Access operand Write-back operand	.89	.54	1.50
47	-(PC)	0	1	1	Decrement PC Access operand Write-back operand	1.78	.54	1.50
50-56	@-(R)	0	2	1	Decrement R Access addr of operand Access operand Write-back operand	1.33	.72	2.10
57	@-(PC)	0	2	1	Decrement PC Access addr of operand Access operand Write-back operand	2.22	.72	2.10
60-67	INDEX(R)	1	1	1	Access index word Sum with R Access operand Write-back operand	1.11	.90	2.10
70-77	@INDEX(R)	1	2	1	Access index word Sum with R Access addr of operand Access operand Write-back operand	1.55	1.08	2.70

*0 if source mode also equals 00-07.

Fetch and Execution Timing

Operand access and update has already been described. The rest of the work performed by an instruction consists of the opcode fetch; the actual data manipulation(s); and any implicit reads or writes.

The table below documents the number of microcycles required times the basic clock time of the processor. Thus, the table directly lists the duration of the fetch and execution phase in microseconds.

The table also indicates the number of implicit data reads and writes that must be performed to complete execution of the operation, including the opcode fetch.

	Execution Time on			Plus	
Opcode	**11/84**	**11/44**	**11/24**	**Reads**	**Write**
Single operand:					
ADC(B)	.22	.54	1.20	1	0
ASR(B)	.22	.54	1.20	1	0
ASL(B)	.22	.54	1.20	1	0
CLR(B)	.22	.54	1.20	1	0
COM(B)	.22	.54	1.20	1	0
DEC(B)	.22	.54	1.20	1	0
INC(B)	.22	.54	1.20	1	0
NEG(B)	.22	.72	1.20	1	0
ROL(B)	.22	.54	1.20	1	0
ROR(B)	.22	.54	1.20	1	0
TST(B)	.22	.54	1.20	1	0
SBC(B)	.22	.54	1.20	1	0
SBC	.22	.54	1.20	1	0
SWAB	.22	.54	1.20	1	0
SXT	.22	.54	1.20	1	0
Double operand:					
ADD	.22	.54	1.20	1	0
BIC(B)	.22	.54	1.20	1	0
BIS(B)	.22	.54	1.20	1	0
BIT(B)	.22	.54	1.20	1	0
CMP(B)	.22	.54	1.20	1	0
MOV(B)	.22	.54	1.20	1	0
SUB	.22	.54	1.20	1	0

Opcode	Execution Time on			Plus	
	11/84	11/44	11/24	Reads	Write
EIS:					
ASH[1]	.88+.22		4.2–30.3	1	0
ASHC[1]	1.01+.22		6.3–46.5	1	0
DIV	7.48		39.6–50.1	1	0
MUL	4.84		23.4–24.0	1	0
XOR	.22	.72	1.20	1	0

NOTE

[1]The execution time of the ASH and ASHC instructions depends on the number of shifts performed. The table states the minimum time (i.e., for 0 bits shifted) plus the incremental time per bit shifted, or the minimum-maximum times.

Opcode	Execution Time on					Plus	
	11/84[2]		11/44[2]		11/24	Reads[2]	Write
Branch:	no	yes	no	yes			
BCC[2]	.44	.88	.36	.90	1.20	1	0
BCS[2]	.44	.88	.36	.90	1.20	1	0
BEQ[2]	.44	.88	.36	.90	1.20	1	0
BGE[2]	.44	.88	.36	.90	1.20	1	0
BGT[2]	.44	.88	.36	.90	1.20	1	0
BHI[2]	.44	.88	.36	.90	1.20	1	0
BHIS[2]	.44	.88	.36	.90	1.20	1	0
BMI[2]	.44	.88	.36	.90	1.20	1	0
BNE[2]	.44	.88	.36	.90	1.20	1	0
BPL[2]	.44	.88	.36	.90	1.20	1	0
BR[2]	.44	.88	.36	.90	1.20	1	0
BVC[2]	.44	.88	.36	.90	1.20	1	0
BVS[2]	.44	.88	.36	.90	1.20	1	0
BLE[2]	.44	.88	.36	.90	1.20	1	0
BLO[2]	.44	.88	.36	.90	1.20	1	0
BLOS[2]	.44	.88	.36	.90	1.20	1	0
BLT[2]	.44	.88	.36	.90	1.20	1	0
SOB[2]	.66	1.11	.72	1.08	1.80–2.10	1	0

NOTE

[2]The execution time for the branch instructions varies depending on whether or not the branch is taken. The table states the values for branch-not-taken and

branch-taken, respectively. In addition, on the 11/84, taking the branch flushes the instruction buffer, requiring an extra read to refill the instruction buffer.

	Execution Time on			Plus	
Opcode	**11/84**	**11/44**	**11/24**	**Reads**	**Write**
Condition codes:					
CLx[3]	.67	.72	2.10	1	0
SEx[3]	.67	.72	2.10	1	0

NOTE

[3]On a given processor, all cases of the CLx and SEx condition code operators execute in the same amount of time.

	Execution Time on			Plus	
Opcode	**11/84**	**11/44**	**11/24**	**Reads**	**Write**
Jump and subroutine:					
JMP	0[4]	0[4]	0[4]	0[4]	0[4]
JSR	0[5]	0[5]	0[5]	0[5]	0[5]
RTS	1.11[6]	1.08	2.10	2[7]	0
MARK	2.22	1.98	3.60	2[7]	0

NOTES

[4]Special case—see JMP Destination Table.
[5]Special case—see JSR Destination Table.
[6]1.33 if other than RTS, PC.
[7]On the 11/84, these instructions flush the instruction buffer, requiring an extra read to refill the instruction buffer.

	Execution Time on			Plus	
Opcode	**11/84**	**11/44**	**11/24**	**Reads**	**Write**
Trap and interrupt:					
BPT	4.44	2.70	6.00	3[8]	2
EMT	4.44	2.70	5.10	3[8]	2
IOT	4.44	2.79	6.00	3[8]	2
RTI	1.98	1.98	3.60	3[8]	2
RTT	1.98	1.98	3.60	3[8]	2
TRAP	4.44	2.70	5.10	3[8]	2

NOTE

[8]On the 11/84, these instructions flush the instruction buffer, requiring an extra read to refill the instruction buffer.

	Execution Time on			Plus	
Opcode	**11/84**	**11/44**	**11/24**	**Reads**	**Write**
Miscellaneous					
CSM	6.23		N/A	2–3	3
HALT		**	6.30–7.50	*	*
MFPI	1.10	1.08	3.6	1	1
MFPD	1.10	1.08	3.6	1	1
MFPS	.22	N/A	1.2	1	0
MFPT	.44	.72		1	0
MTPI	.66	1.26	1.8	2	0
MTPD	.66	1.26	1.8	2	0
MTPS	1.78	N/A	4.2	1	0
NOP	.66	.72	2.10	1	0
RESET	≈ 10,000[10]	≈ 100,000[10]	≈ 100,000[10]	1	0
SPL	1.55	2.16	N/A	1	0
TSTSET	1.11	N/A	N/A	1	1
WAIT[9]	−∞	.72–∞	2.4–∞	1	0
WRTLOK	.88	N/A	N/A	1	1

NOTE

[9]The value stated in the table is the minimum execution time for the WAIT instruction. WAIT actually continues executing until the next interrupt request is received.

[10]The RESET instruction timing is actually determined by an R/C one-shot.
**The HALT instruction is not microcoded on the 11/44.
*Varies for the 11/84, 11/44 and 11/24.

• Floating-Point Instructions

The same basic methodology can be applied to the floating point instructions, with a few additional concerns added

Operand Length
PDP-11 floating-point operands may be 32- or 64-bits in length. In addition, mode 27 operands result in the access of just 16 bits. This means that one, two, or four words must be read or written from or to memory to access or write floating-point operands. This obviously affects the execution time of the floating-point instructions.

Data Dependencies
Most of the basic PDP-11 instructions require a fixed number of microcycles to execute. ASH and ASHC (the long-shift instructions) vary in the number of microcycles based on the number of bits shifted, while BRANCH instructions vary slightly depending on whether or not the branch is taken.

On the other hand, the number of microcycles required to execute a floating-point instruction can vary greatly depending on the specific floating-point data used.

This is due to two facts:

- Before two floating-point numbers can be added or subtracted, their radix points must be aligned.
- Before a floating-point number can be stored, it must be normalized (so that its mantissa is in the range of 0.5 to 0.999...).

Both of these operations require long shifts of varying lengths. This affects the execution time of the instructions.

• PDP-11/84 Floating Point Instruction Timing: FPJ11

Because the FPJ11 is a coprocessor operating in parallel with the J-11 chipset, the calculation of floating point instruction times (for J-11 systems that utilize the FPJ11) must take this parallel processing into account.

TERM	DEFINITION
FPJ11 cycle	Two clock periods (110 ns at 18 MHz)
J-ll nonstretched cycle	Two FPJ11 cycles (220 ns at 18 MHz)

TERM	DEFINITION
J-11 read cycle	J-11 nonstretched cycle if cache hit. Dependent on read access time of the system if cache miss, the minimum is two J-11 nonstretched cycles, after which the J-11 stretches in 1/2 cycle increments until MCONT is asserted.
J-11 write cycle	Dependent on write access time of the system (Two J-11 cycles + 1/2 cycles until MCONT).
Instruction Decode	A decode/prefetch cycle followed by a MOV microinstruction that allows the FPJ11 to assert DMR prior to the start of the next microinstruction (INPR for REG mode). This time equals two nonstretched cycles if the prefetch is a cache hit, else one nonstretched cycle plus one read cycle.
Address Calculation Time	J-11 time required to calculate the address of the operand. This time is dependent on the addressing mode of the instruction, the frequency of the system clock, and whether any indirect data required is present in the cache. See Table C-1.
Argument Transfer Time	J-11 time required to load or store floating-point operands. This time is one nonstretched cycle (address relocation microcycle) plus one read cycle per 16-bit word read from memory for load class instructions—or one nonstretched cycle plus one write cycle per 16-bit word written to memory for store class instructions.
INPR (FEATEMP, TEMP)	J-11 support code microinstruction executes for all FPJ11 instructions. It moves the PC of the previous FPJ11 instruction to a TEMP register in case that instruction resulted in a floating-point exception. If the FPJ11 is still executing the previous floating-point instruction when the J-11 reaches its INPR microinstruction, the FPJ11 asserts STALL causing the J-11 INPR microcycle to stretch. The J-11 then WAITs for the FPJ11 to deassert STALL, signaling the system interface to assert MCONT before executing the next microinstruction (OUTR).

TERM	DEFINITION
WAIT	J-11 time waiting for the completion by the FPJ11 of the previous floating-point instruction. For load class or REG mode instructions, this time is from when the J-11 INPR cycle stretches at the trailing edge of MALE until the FPJ11 deasserts STALL. This time equals zero if a stall was not required or if the FPJ11 deasserted the STALL signal after the INPR cycle began, but prior to the trailing edge of MALE. Although the WAIT time for this latter case is zero, RESYNC time is required. For store class instructions the WAIT time equals the time between the assertion of SCTL (i.e., when the system interface is ready to execute the first write cycle of a floating-point store) and the assertion of FPA-RDY (data ready) by the FPJ11.
RESYNC	For load class and REG mode instructions, this is the time required to continue a stretched INPR. This is the time for the SYSTEM INTERFACE to recognize the deassertion of STALL and assert MCONT, plus the time required for the J-11 to synchronize MCONT and advance to the next microinstruction. Store class instructions normally do not have RSYNC time because the J-11 is waiting in a stretched write cycle and the continuation time is part of the write cycle. However, if the FPJ11 is executing a previous MODF/D or DIVD, the FPJ11 will assert STALL in order to stretch a non-I/O cycle prior to the first bus write. This allows the SYSTEM INTERFACE to service DMA thus limiting the worst case DMA latency when waiting for FPJ11 output. In this case a WAIT and RESYNC time associated with the stretched non-I/O cycle is added to the effective execution time of the store class instruction.
OUTR (PC,FEATEMP), TESTPLA FPE	Last J-11 support microinstruction unless there is a floating-point execution from the previous floating-point instruction. This saves the address of PC in FEATEMP.

TERM	DEFINITION
PRDC SYNC	Time required by FPJ11 to decode a floating-point instruction and begin execution after receiving PRDC. This time equals two or three FPJ11 cycles depending on synchronization. PRDC SYNC is not added to FPJ11 instruction execution times when the FPJ11 is executing a previous floating-point instruction at the assertion of PRDC.
Floating-Point Execution Time	Time required by FPJ11 to complete a floating-point instruction once it has received all arguments. For store class instructions, floating-point execution time includes the time from the start of the instruction until the FPJ11 asserts FPA-RDY indicating the first 16-bit word is available for output. See Table C-2.
Effective Execution Time	Total J-11 time required to execute a floating-point instuction.
Load class	Instruction Decode + Address Calculation + Argument Transfer + INPR + WAIT + RSYNC + OUTR.
REG mode	Instruction Decode + INPR + WAIT + RSYNC + OUTR.
Store class	Instruction Decode + Address Calculation + INPR + Argument Transfer + WAIT + OUTR.

Load class instructions require input data and deposit results to the destination floating-point accumulator. REG mode instructions are floating-point accumulator to floating-point accumulator.

Execution of a load class floating-point instruction by the FPJ11 occurs in parallel with J-11 operation and can be overlapped. See Figure C-1.

Store class instructions can be overlapped by the J-11 as the FPJ11 will complete a previously started load class or REG mode instruction and then continue to store the instruction. Execution of the store class instruction must be completed before the result can be stored to memory, thus eliminating further parallel processing for store class floating-point instructions. See Figure C-2.

Table C-1 ▪ Address Calculation Times

MODE	LOAD CLASS	STORE CLASS
0	0	0
1	3	3
2	3	2
3	3+RD	2+RD
4	4	4
5	3+RD	3+RD
6	3+RD	2+RDI
7	3+RDI+RD	3+RDI+RD
27	2	2
37	2+RDI	1+RDI
67	2+RDI	2+RDI
77	4+RDI+RD	4+RDI+RD

RDI=J-11 I-stream request
RD =J-11 Read cycle

Table C-2 ▪ FPJ11 Instruction Times (18MHz = 111ns cycle)

INSTRUCTION	MIN CYCLES	TYP CYCLES	MAX CYCLES	STRETCH CYCLES	18 MHz TYP (μs)
ADDF/SUBF	7	9	19	5	1.0
ADDD/SUBD	7	9	30	5	1.0
MULF	15	15	16	11	1.7
MULD	26	26	27	22	2.9
DIVF	17	24	30	25	2.7
DIVD	33	48	62	57	5.4
MODF	28	34	43	15	3.7
MODD	39	45	71	26	5.0
CMPF/D	3	4	6	2	.4
LDF/D	3	3	3	0	.3
LDEXP	2	2	2	0	.2
LDCIF/D	10	10	10	3	1.1
LDCLF/D	10	10	10	3	1.1
LDCFD	4	4	4	1	.4
LDCDF	4	4	8	1	.4
STF/D	3	3	3	0	.3
STCFI	8	10	13	1	1.1
STCFL	8	12	16	1	1.3
STCFD	4	4	4	0	.4
STCDF	6	6	6	1	.7
STEXP	5	5	5	0	.6
TSTF/D,LDFPS STFPS,CFCC,SET	3	3	3	0	.3
ABSF/D,NEGF/D	4	4	5	0	.4

NOTE

Stretch cycles indicate the number of cycles out of maximum cycles that a data dependent stretch of one additional cycle could occur with probability less than 1% for each additional cycle.

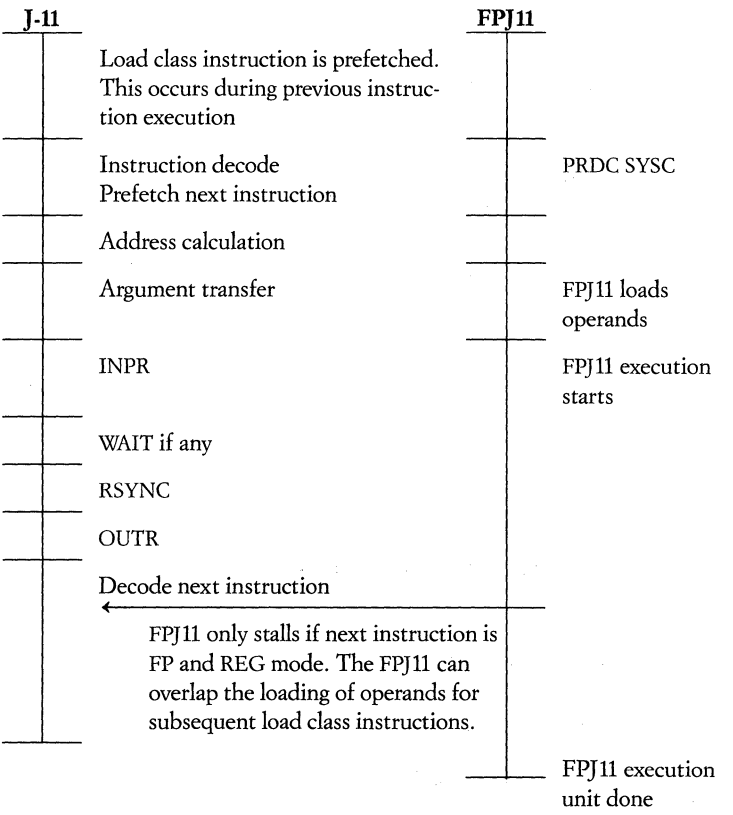


Figure C-1 ▪ J11/FPJ11 Interaction for Load Class Instructions

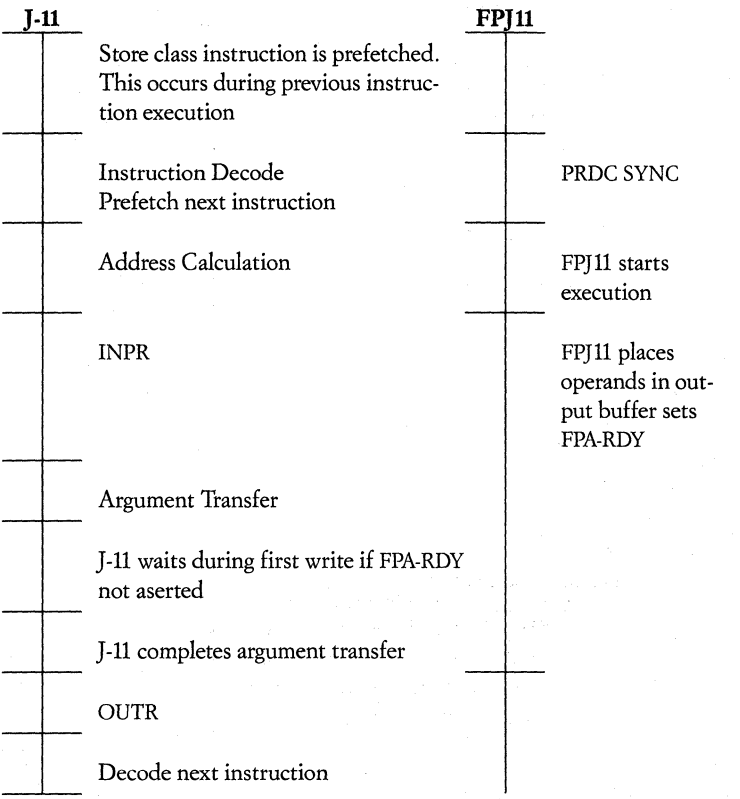


Figure C-2 ▪ J-11/FPJ11 Interaction for Store Class Instructions

• PDP-11/44 Floating Point Instruction Timing: FP11-F

Instruction Execution Time

The execution time of an FP11-F floating point instruction is dependent on the following conditions

- type of instruction.
- type of addressing mode specified.
- type of memory.
- memory management facility enabled or disabled.

Additionally, the execution time of certain instructions, such as ADD, is dependent on the data.

Table C-3 provides the basic instruction times for mode 0. Tables C-4 through C-8 show the additional time required for instructions other than mode 0. For example, to calculate the execution time of a MULF (single-precision multiply) for mode 3 (autoincrement deferred) with the result to be rounded:

1. Refer to Table C-3 which gives MULF, mode 0, execution time of 12.4 μs.
2. Refer to Note 1 as specified in the notes column of Table C-3. Note 1 specified an additional 0.84 μs is to be added if rounding mode is specified. This yields 12.24 μs.
3. The Modes 1 through 7 column of Table C-3 refers to Table C-4 to determine the additional time required for mode 1 through 7 instructions. In this example, mode 3 specifies an additional 3 μs for single precision yielding 16.24 μs.

All timing information is in microseconds unless otherwise noted. Times are typical; processor timing can vary ± 10%. All instructions assume 100% cache hits.

NOTE

Add .090 μs for each DATI memory cycle if memory management is enabled. Add .630 μs for each DATI memory cycle if a cache miss is encountered.

Table C-3 ▪ FP11-F Instruction Execution Times

Instruction	Mode 0 (Reg. to Reg.)	Notes	Modes 1 thru 7
LDF	3.0		
LDD	3.0		
LDCFD	4.8	1	
LDCDF	4.8	1	
CMPF	4.5		
CMPD	4.5		
DIVF	12.3	1	
DIVD	19.6	1	Use Table C-4 to
ADDF	6.5	1,2	determine memory-
ADDD	6.5	1,2	to-register times
SUBF	6.9	1,2	for these
SUBD	6.9	1,2	instructions
MULF	12.4	1	
MULD	19.7	1	
MODF	16.4	1,3	
MODD	23.7	1,3	
STF	1.4		Use Table C-5 to
STD	1.4		determine memory-
STCDF	4.2		to-register times
STCFD	4.2		for these
CLRF	1.6		instructions
CLRD	1.6		
ABSF	2.5		Use Table C-6 to
ABSD	2.5		determine memory-
NEGF	2.6		to-memory times
NEGD	2.6		for these
TSTF	2.6		instructions
TSTD	2.6		
LDFPS	1.5		Use Table C-7 to
LDEXP	3.4		determine memory-
LDCIF	6.5	1,4	to-register times
LDCID	6.5	1,4	for these
LDCLF	6.5	1,4	instructions
LDCLD	6.5	1,4	
STFPS	1.8		Use Table C-8
STST	1.6		to determine
STEXP	2.4	5	register-to-memory

Instruction	Mode 0 (Reg. to Reg.)	Notes	Modes 1 thru 7
STCFI	3.5	5	times for these
STCDI	3.5	5	instructions
STCFL	3.5	5	
STCDL	3.5		
The following instructions do not reference memory			
CFCC	1.0		
SETF	1.2		
SETD	1.2		Execution times
SETI	1.2		are as shown.
SETL	1.2		

Table C-4 • Floating Source Fetch Time

	Memory Cycles		Time (μs)	
Addressing Mode	Single Precision	Double Precision	Single Precision	Double Precision
1	2	4	0.60	1.4
2	2	4	0.80	1.6
2 Immediate	1	1	0.30	0.3
3	3	5	0.90	1.7
4	2	4	0.80	1.6
5	3	5	0.90	1.7
6	3	5	1.10	1.9
7	4	6	1.40	2.2

Table C-5 • Floating Destination Store Time

	Memory Cycles		Time (μs)	
Addressing Mode	Single Precision	Double Precision	Single Precision	Double Precision
1	2	4	0.38	2.94
2	2	4	1.56	3.12
2 Immediate	1	1	0.60	0.60
3	3	5	1.68	3.24
4	2	4	1.56	3.12
5	3	5	1.68	3.24
6	3	5	1.86	3.42
7	4	6	2.16	3.72

Table C-6 ▪ Floating Destination Fetch And Store Time

Addressing Mode	Memory Cycles		Time (μs)	
	Single Precision	Double Precision	Single Precision	Double Precision
1	2	2	0.72	0.72
2	2	2	0.90	0.90
2 Immediate	2	2	0.80	0.80
3	3	3	1.02	1.02
4	2	2	0.90	0.90
5	3	3	1.20	1.20
6	3	3	1.20	1.20
7	4	4	1.50	1.50

Table C-7 ▪ Source Fetch Time

Addressing Mode	Memory Cycles		Time (μs)	
	Short Integer	Long Integer	Short Integer	Long Integer
1	1	2	0.30	0.70
2	1	2	0.48	1.28
2 Immediate	1	1	0.48	0.48
3	2	3	0.60	1.00
4	1	2	0.48	1.28
5	2	3	0.60	1.00
6	2	3	0.78	1.18
7	3	4	1.08	1.48

Table C-8 ▪ Destination Store Time

Addressing Mode	Memory Cycles		Time (μs)	
	Short Integer	Long Integer	Short Integer	Long Integer
1	1	2	0.60	1.38
2	1	2	0.96	1.68
2 Immediate	1	1	0.96	0.96
3	2	3	0.90	1.68
4	1	2	0.96	1.68
5	2	3	0.90	1.68
6	2	3	1.08	1.86
7	3	4	1.38	2.16

NOTES

1. Add 0.84 μs when in rounding mode (FT = 0).
2. Add 0.24 μs per shift to align binary points and 0.24 μs per shift for normalization. The number of alignment shifts is equal to the exponent differnces bounded as follows

   $1 \leq$ EXP(AC)—EXP(FSRC) ≤ 24, single precision.
   $1 \leq$ EXP(AC)—EXP(FSRC) ≤ 56, double precision.

   The number of shifts required for normalization is equivalent to the number of leading zeros of the result.
3. Add 0.24 μs times the exponent of the product if the exponent of the product is

   $1 \leq$ EXP(PRODUCT) ≤ 24, single precision.
   $1 \leq$ EXP(PRODUCT) ≤ 56, double precision.

   Add 0.24 μs per shift for normalization of the fractional result. The number of shifts required for normalization is equivalent to the number of leading zeros in the fractional result.
4. Add 0.24 μs per shift for normalization of the integer being converted to a floating point number. For positive integers, the number of shifts required to normalize is equivalent to the number of leading zeros; for negative integers, the number of shifts required for normalization is equivalent to the number of leading ones.
5. Add 0.24 μs per shift to convert the fraction and exponent to the integer form, when the number of shifts is equivalent to 16 minus the exponent when converting to short integer, or 32 minus the exponent when converting to long integer for exponents bounded as follows

   $1 \leq$ EXP(AC) ≤ 15, short integer.
   $1 \leq$ EXP(AC) ≤ 31, long integer.

▪ PDP-11/24 Floating-Point Instruction Timing: KEF11-AA

Floating-point instruction timing for the PDP-11/24, using the KEF11-AA chipset, may be computed using Tables C-9 through C-16. The following conditions must be accounted for in making these calculations.

- Type of instruction—each floating-point instruction is listed in Table C-9 along with instruction fetch and execution times for addressing mode zero. Each table is divided into single precision and double precision catagories.
- Type of addressing—when using any mode other than zero, Tables C-10 through C-14 must be referenced for additional time necessary for transfers between memory and FPP registers.

- Special conditions—conditions that affect execution time are listed in Table C-15. If an instruction lists any of these conditions in the "Notes" column of Table C-9, each specified note must be referenced and the appropriate times added.
- State of memory management if memory management is enabled, the time for each DATI/DATIP is increased by 75 nanoseconds and the time for each DATO/DATOB is increased by 150 nanoseconds.
- Type of memory—these times were calculated using the MOS MS11-L memory with typical access times of 385 nanoseconds for DATI/DATIP and 125 nanoseconds for DATO/DATOB. When using other memories, any difference in access times must be accounted for.

All times are typical and may vary by ± 10 percent.

Table C-9 ▪ Instruction Fetch and Execution Times

Instruction	Microcycles	Mode 0 (μs)	Notes	Modes 1-7
ADDF	119	39.22	1-9	
ADDD	135	44.78	1-9	
SUBF	122	40.13	1-9	
SUBD	138	45.68	1-9	
MULF	168	53.85	8-11,13	
MULD	641	196.50	8,9,12,13	
MODF	229	72.98	8-11,20-26	For these
MODD	694	213.23	8,9,12,20-26	instructions use
DIVF	302	94.05	8,9,13	Table C-10 to
DIVD	795	242.70	8,9,13	determine
CMPF	65	22.58	14,15	additional times
CMPD	71	24.83	14,15	due to
TSTF	30	10.35	9	memory-to-register
TSTD	34	11.85	9	transfer.
LDF	29	10.20	9	
LDD	37	13.20	9	
LDCFD	41	14.03	9	
LDCDF	56	19.05	9	
CRLF	37	12.45	9	For these instructions use
CLRD	41	13.95	9	Table C-11 to
STF	19	6.90	—	determine
STD	27	9.90	—	additional times
STCFD	47	15.98	9	due to
STCDF	66	22.35	9	memory-to-register transfer.

Instruction	Microcycle	Mode 0 (μs)	Notes	Modes 1–7
ABSF,NEGF	43	14.48	9	For these instructions use Table C-12 to determine additional times due to memory-to-register transfer.
ABSD,NEGD	51	17.48	9	
LDFPS	16	5.63	—	For these instructions use Table C-13 to determine additional times due to memory-to-register transfer.
LDEXP	40	13.35	9	
LDCIF,LDCLF	59	19.28	16,17	
LDCID,LDCLD	54	17.85	16,17	
STFPS	13	4.73	—	For these instructions use Table C-14 to determine additional times due to memory-to-register transfer.
STEXP	36	12.08	9	
STCFI	58	18.75	9,18,19,27,28	
STCFL	57	18.45	9,18,19,27,28	
STCDI	59	19.20	9,18,19,27,28	
STCDL	58	18.90	9,18,19,27,28	
STST	18	6.38	—	
CFCC	13	4.73	—	For these instructions memory is not referenced.
SETF,SETD, SETI,SETL	15	5.40	—	

Table C-10 ▪ Floating Source Fetch Time*

	Memory Cycles		Time (μs)	
Mode	Single Precision	Double Precision	Single Precision	Double Precision
1	2	4	2.55	4.20
2	2	4	2.85	4.50
2 Immed	1	1	2.10	0.60
3	3	5	3.45	5.10
4	2	4	3.15	4.80
5	3	5	3.75	5.40
6	3	5	3.75	5.40
7	4	6	4.95	6.60

*If floating source is negative, add 0.60 microseconds.

Table C-11 ▪ Floating Destination Store Time

	Memory Cycles		Time (μs)	
Mode	Single Precision	Double Precision	Single Precision	Double Precision
1	2	4	1.60	2.90
2	2	4	2.58	3.88
2 Immed	1	1	-0.40	-1.90
3	3	5	2.50	3.80
4	2	4	2.88	4.18
5	3	5	2.80	4.10
6	3	5	2.80	4.10
7	4	6	4.00	5.30

Table C-12 ▪ Floating Destination Fetch and Store Time

	Memory Cycles		Time (μs)	
Mode	Single Precision	Double Precision	Single Precision	Double Precision
1	4	8	4.15	7.10
2	4	8	5.43	8.38
2 Immed	2	2	1.70	-1.30
3	6	10	5.95	8.90
4	4	8	6.03	8.98
5	6	10	6.55	9.50
6	6	10	6.55	9.50
7	8	12	8.95	11.90

Table C-13 • Source Fetch Time

Mode	Memory Cycles		Time (μs)	
	Single Precision	Double Precision	Single Precision	Double Precision
1	1	2	1.20	2.40
2	1	2	1.50	2.70
2 Immed	1	1	0.90	0.90
3	2	3	2.10	3.30
4	1	2	1.80	3.00
5	2	3	2.40	3.60
6	2	3	2.40	3.60
7	3	4	3.60	4.80

Table C-14 • Destination Store Time

Mode	Memory Cycles		Time (μs)	
	Single Precision	Double Precision	Single Precision	Double Precision
1	1	2	1.03	2.05
2	1	2	1.33	2.35
2 Immed	1	1	0.73	0.73
3	2	3	1.93	2.95
4	1	2	1.63	2.65
5	2	3	2.23	3.25
6	2	3	2.23	3.25
7	3	4	3.43	4.45

Table C-15 • Notes on Special Conditions

Condition	Time (μs)	
	Single Precision	Double Precision
1. Addition of opposite signs or subtraction of like signs.	0.60	1.20
2. Alignment of binary points is required (n = exponent difference).	(1.20)n	(1.80)n
3. Exponent difference ≷ 7 but not = 0.	1.20	1.20
4. Exponent difference > 26.	-2.10	-2.10
5. ACC exponent < SRC exponent.	1.95	3.30

6. ACC exponent = SRC exponent.	see Table 8	see Table 8
7. Normalization is required (n = number of shifts necessary to normalize).	0.30 + (1.20)n	0.30 + (1.80)n
8. Rounding is selected.	1.20	1.80
9. Result is positive.	0.30	0.30
10. DST < 15-00 > not = 0.	29.10	—
11. DST < 22-00 > not = 0 (n = number of bits in DST < 22-00 > which are set.	(0.60)n	—
12. DST < 54-00 > not = 0 (n = number of bits in DST < 54-00 > which are set.	—	(1.20)n
13. Normalization is required.	0.60	1.20
14. Both ACC & SRC have same sign, but are not equal.	1.80	3.15
15. ACC = SRC	0.90	2.25
16. The integer to be converted $< 2^{15}$ (single precision) or $< 2^{31}$ (double precision) (n = number of leading zeros in binary representation of the integer, excluding the sign bit.	(1.20)n	(1.20)n
17. Result is negative.	0.30	0.90
18. Unbiased exponent > 15.	0.90	0.90
19. Unbiased exponent not equal to 8 or 24 (if exponent > 15, n = [expn-24]) (if exponent ≤ 15, n = [expn-81]).	(2.10)n	(2.10)n
20. Normalization of product is required prior to separation of integer and fraction.	1.20	1.20
21. Normalization of product is required after separation of integer and fraction (n = number of shifts necessary to normalize).	0.38 + (1.80n)	0.38 + (1.80n)
22. The number represented by the low 4 bits of the product exponent not equal to 0 (n = number represented by these 4 bits).	(0.90)n	(0.90)n
23. Biased exponent ≧ 16 but < 32.	0.30	0.30
24. Biased exponent ≧ 32 but < 48.	0.60	0.60

25. Biased exponent ≧48 but <64.	0.90	0.90
26. Biased exponent ≧64.	0.60	0.60
27. Resulting integer is negative.	0.98	0.98
28. Result is a long integer.	0.30	0.30

Numbers referred to in the text of the notes are decimal unless otherwise specified.

Table C-16 • Additional Time Under Condition [ACC Expn = SRC Expn] (see Note 6)

Single Precision	
Condition	**Time (μs)**
ACC<22-16> > SRC<22-16>	0.60
ACC<22-16> < SRC<22-16>	2.55
ACC<15-00> > SRC<15-00>*	1.20
ACC<15-00> < SRC<15-00>*	3.15
ACC<22-00> = SRC<22-00>	0.60
Double Precision	
Condition	**Time (μs)**
ACC<54-48> > SRC<54-48>	0.60
ACC<54-48> > SRC<54-48>	3.90
ACC<47-32> > SRC<47-32>*	1.20
ACC<47-32> > SRC<47-32>*	4.50
ACC<31-16> > SRC<31-16>*	1.80
ACC<31-16> > SRC<31-16>*	5.10
ACC<15-00> > SRC<15-00>*	2.40
ACC<15-00> > SRC<15-00>*	5.70
ACC<54-00> = SRC<54-00>	1.80

*All higher bits of ACC and SRC mantissas are equal.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W

X

Y

PDP-11 UNIBUS Processor Handbook

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our handbooks.

What is your general reaction to this handbook? (format, accuracy, completeness, organization, etc.)

_____

_____

_____

What features are most useful? _____

_____

_____

Does the publication satisfy your needs? _____

_____

_____

What errors have you found? _____

_____

_____

_____

Additional comments _____

_____

_____

Name _____

Title _____

Company _____ Dept. _____

Address _____

City _____ State _____ Zip _____

EB-26077-41

digital

(staple here)

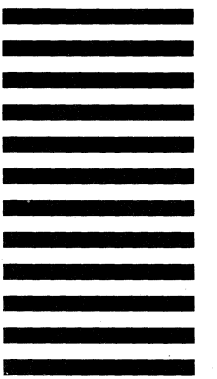
(please fold here)

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD, MASS.

POSTAGE WILL BE PAID BY ADDRESSEE



Digital Equipment Corporation
Corporate Communications Services
CFO 1-2/M92
200 Baker Avenue
West Concord, MA 01742

digital