digital

VAX Hardware Handbook

Volume 1–1986

digital

VAX Hardware Handbook

Volume 1–1986

VAX Hardware Handbook
Volume 1–1986

digital

Contents

Chapter 1 ▪ Introduction

Preface

Through the years Digital's VAX system hardware and software have proven to be a valuable asset to businesses and institutions of all sizes. The versatility and reliability of VAX systems, coupled with a worldwide network of dependable training and service facilities, has produced many satisfied customers. The continuing improvements in software functionality and hardware design have opened many new areas for VAX system application. VAX systems are available for small dedicated applications in office areas, laboratories, and manufacturing sites, and for large distributed data processing applications where data and processing power is shared between many systems over a wide geographical area.

Digital's consistency of purpose is to design VAX systems that are both hardware compatible and software compatible and to provide a comprehensive network by which these systems and other manufacturers' systems can effectively communicate to share resources and processing power. By adhering to the standards of the VAX hardware and software architecture, programs can be developed on the large VAX systems and for operation on the smaller VAX systems. Application programs developed for one system can be migrated to other VAX systems without revisions.

In addition to the general purpose VMS operating systems and extensive layered software that has provided VAX users with a reliable, high-performance software environment for many years, most of the VAX systems now operate with the ULTRIX-32 operating systems. ULTRIX-32 software is Digital's supported native UNIX™ operating system that provides complete UNIX functionality enhanced especially for the VAX systems.

Digital has recently developed many new hardware products to extend the capabilities of existing VAX systems and to allow VAX processors to be used for new applications. Several new processors and terminals have been added to satisfy the need for additional small, medium, and large systems. High-capacity, mass-storage devices and intelligent controllers now provide the user with the solution to the ever-expanding data storage requirements. New communication products have been developed to enhance the existing product line and allow the efficient expansion of network capabilities. A high-speed computer interconnect was designed to permit the integration of VAX processors, intelligent mass storage facilities and communication networks into VAXcluster configurations.

Digital Network Architecture (DNA) is a comprehensive framework of hardware and software products to satisfy all communication needs for Local Area Networks (LANs) and distributed data processing. These products include DECnet, Ethernet, SNA Gateway and a series of communication servers.The continued development of communications products ensures efficient information transfers between systems and devices.

Digital's Storage Architecture (DSA) is a combination of hardware and software that complements the new and existing VAX systems by enabling intelligent mass-storage controllers to perform many of the data storage functions usually assigned to the processor. This sophisticated system increases data integrity and security and provides faster and more flexible access to the stored information.

VAX Hardware Handbook

This handbook provides general technical information for the VAX hardware product line to help you evaluate your system requirements. It includes descriptions and specifications for the VAX processors, data storage systems and devices, VAXcluster configurations and communication products. Additional information on the VAX system hardware is available from Digital sales offices. This handbook is part of the VAX handbook set which includes the following:

VAX Architecture Handbook – Detailed descriptions of the VAX architecture including virtual address, data representations, instruction formats, addressing modes, interrupt schemes, and memory management.

VAX Software Handbook – Description of the software capabilities of VAX/VMS operating systems, the Digital Command Language (DCL), programming languages, and data communications.

Chapter 1 · Introduction

The VAX system family is composed of Digital's 32-bit VAX processors and devices that are used for a wide range of applications in the technical, commercial, academic, and research environments. More than 30,000 VAX systems installed throughout the world have proven VAX computing to be a reliable and efficient system for a variety of applications. Because of the many types of VAX processors and devices that are available, the VAX systems can be tailored for specific applications and to conform to many budgets constraints. All VAX system family members are compatible with each other because they include the proven VAX architecture and operate with the same VMS virtual memory operating system. Software developed for one VAX system performs equally well on any other VAX system. Systems can be easily expanded to fulfill increased processing requirements without the cost of adding noncompatible software and hardware.

The new members of the VAX system family include the MicroVAX I, a low-cost, small and compact microcomputer developed for dedicated applications; the VAX-11/725 system, which is a low-noise, specially configured version of the VAX-11/730 system designed specifically for the open-office environment; the VAX-11/785 system, which is a more powerful single-processor system than the VAX-11/780 system; and the VAX 8600, a high-performance system that provides the speed and power needed for large-scale, single and multiuser processing applications.

In some of the chapters of this handbook, the VAX processors are referred to as small, medium, and large, based on their size and processing power. These references are defined as follows unless otherwise noted in the individual chapters:

Reference	VAX Processor
Small	VAX-11/725 and VAX-11/730
Medium	VAX-11/750
Large	VAX-11/780, VAX-11/782, VAX-11/785, and VAX 8600

· VAX Processor Features

All VAX processors implement the 32-bit VAX architecture, an extensive instruction set with numerous data types, and a 32-bit bus structure for high data throughput. This is coupled with a virtual address space of up to 4 Gbytes, sixteen 32-bit general purpose registers, 12 addressing modes, and 32 interrupt levels–which together provide a versatile and efficient processor system.

Common to all VAX systems is a microprocessor-based console subsystem that allows the user, system manager, or service engineers to communicate with the system through the console terminal. In console mode, the processor can be started or halted, self-tested, initialized to a known state, and single-stepped through instructions. Information in memory, storage locations, and internal registers can be examined and data can be deposited into these locations. Diagnostic maintenance and bootstrap programs can be loaded from the console load device and controlled by the console terminal. When the remote diagnostic service is included as part of a Digital Field Service contract, the diagnostic testing can also be controlled through the console subsystem from a remote Digital test facility.

VAX system hardware is complemented by the VMS operating system, which is continually being enhanced to provide more capabilities and more efficient system operation. VMS is a powerful multiprogramming operating system capable of supporting many users in realtime, interactive timesharing, and multistream batch applications. It also provides support for online program development.

The ULTRIX-32 operating system has been added to the VAX system family and provides users with a commodity-software base for 32-bit systems. The ULTRIX-32 system, Digital enhanced native-node UNIX* operating system, offers the standard set of UNIX languages and utilities. When a high level implementation language is desirable for limited or dedicated functions, the VAXELN Toolkit is available for the small and medium VAX processors running on a current VMS software version.

The full capabilities of the VAX hardware and software are further enhanced by the VAXcluster architecture, Digital Network Architecture (DNA), and the Digital Storage Architecture (DSA).

The VAXcluster architecture enables up to 16 VAX processors or HSC50 intelligent mass-storage servers to be connected together through the SC008 Star Coupler unit to form a single high-performance system.

* UNIX is a trademark of AT&T Bell Laboratories

Through the VAXcluster, the processing power can be shared by several VAX processors and each processor is allowed access to data of a common data file. The HSC50 is an intelligent-mass storage I/O server that optimizes the system data throughput by controlling the operation of up to 24 disk drives or tape drives and formatters. The SC008 Star Coupler unit forms the common connection point for the cables from the processors and intelligent servers. By providing common access to data files, the cost of development software and support of multiple databases when a system is expanded is significantly reduced. When more processing power is required, the VAX system selected can be tailored to the new system requirements without adding unnecessary hardware or software.

The DNA enables communication between VAX systems and other Digital systems and between Digital systems and systems developed by other manufacturers. Digital offers extensive capabilities that permit the linking of computers and terminals into flexible configurations to increase the efficiency and cost-effectiveness of data processing operations. Ethernet is Digital's high-speed local area communications network that enables computer systems and terminals, whether located centrally or at remote sites, to exchange information and to share resources.

DSA is Digital's framework of standardized interfaces that permit the addition of new products, and technologies that operate with different host systems without the need to develop new controllers and device drivers. DSA allows an expanding group mass storage products, including disk and magnetic tape drives, to be implemented into an intelligent mass-storage subsystem that provides a high standard of data integrity, fast data throughput, and many reliability features.

VAX System and VAXcluster Building Blocks

The desired hardware configurations of the standard VAX systems and VAXcluster systems are obtained by using the system building block (SBB) menus that are described in the current version of the *VAX Systems and Options Catalog.* The menus allow the user to select a basic system consisting of a VAX CPU, main memory, and software operating system license, and to add the required interfaces and devices. From the system menu, the user can select the system storage and load devices, communication interfaces, console terminals and operating system documentation and software media. Using the system building blocks, VAX systems and VAXcluster configurations can be selected for specific applications without the purchase of unneeded hardware and software. The *VAX Systems and Options Catalog* can be obtained at any Digital sales office or from a Digital sales representative. Figure 1-1 shows a typical system building block diagram for the VAX 8600 VAXcluster system.

Components	Order Code
VAX-11/730 CPU	730XA-AE(AJ)
2-Mbyte ECC MOS memory	
Dual TU58 Cassette	
VAX/VMS license and warranty	

A fully-supported system requires a system device, load device, communications device and console terminal. If your order does not include the appropriate devices, it may not be maintainable. In this case you should contact your local Field Service branch office.

SYSTEM DEVICE	LOAD DEVICE, Disk/Tape (PE = 1600 b/in, GCR = 6,250 b/in)		
	RA60 205 MB (Rem Disk)	**TU80 MAGTAPE (PE)**	**TU81 MAGTAPE (PE,GCR)**
RA80 121 MB (Fixed Disk)	RUA80-AA(AD) RA60-CA(CD)	RUA80- AA(AD) TU80-AA(AB)	RUA81- AA(AD) TU81-AA(AB)
RA60 205 MB (Rem Disk)	RUA60- CA(CD) RA60-AA	RUA60- CA(CD) TU80-AA(AB)	RUA60- CA(CD) TU81-AA(AB)
RA81 456 MB (Fixed Disk)	RUA81- AA(AD) RA60-CA(CD)	RUA81- AA(AD) TU80-AA(AB)	RUA81- AA(AD) TU81-AA(AB)
RA81 1,368 MB (3-Fixed Disks)	RUA81- EA(ED) RA60-CA(CD)	RUA81- EA(ED) TU80-AA(AB)	RUA81- EA(ED) TU81-AA(AB)

Communication Devices	Order Code
Multipurpose communications interface	DMF32-LP
8-line EIA asynchronous serial communications interface	DZ11-DP
8-line 20-mA asynchronous serial communications interface	DZ11-HP
16-line EIA asynchronous serial communications interface with modem control	DHU11-AP
24-line EIA asynchronous serial communications interface with modem control	DMZ32-AP*
24-line EIA asynchronous serial communications interface without modem control	DMZ32-DP*
UNIBUS to Ethernet controller	DEUNA-AA

Console Terminals	Order Code
Hardcopy terminal	LA120-DA
Tabletop terminal	LA100-BA
Stand (option)	LA10X-SL
Tabletop terminal	LA12-DB
Stand (option)	LA12X-SL

Purchase of media and documentation or a System Startup Service Package is required for the first VAX system of each CPU type. System Startup Service Packages include media and documentation for VMS and all dependent products purchased. See Software Services Section. Complete software order codes with a J for RA60 or a M for Magtape.

Software Options	Order Code
VAX/VMS media and documentation (H-kit)	QC001-H
SSSP LEVEL I	QC001-5
SSSP LEVEL II	QC001-7
SSSP LEVEL III	QC001-B

*Remote device only—see Options Section for details

Figure 1-1 • Typical VAX System Building Block Menu

MicroVAX I Microcomputer System

The MicroVAX I microcomputer system is the smallest and least expensive of the VAX systems. It is engineered to perform dedicated realtime functions, including workstation applications, office automation, and small business data processing. Because the MicroVAX uses the extended Q-bus, many inexpensive interface options developed for PDP-11 computer systems can be easily implemented into the MicroVAX I. The MicroVAX I is described in detail in the *MicroVAX I Handbook* (document number EB-25156-47), which is available from any Digital sales office.

VAX-11/725 Open-office System

The VAX-11/725 system is a compact and quiet processor system designed specifically for the open-office environment and for single-user and multiuser work stations. The system operates with the VMS (virtual memory system) and provides the power and reliability of the VAX-11/730 processor in a small pedestal cabinet. The VAX-11/725 has been specially reconfigured to reduce noise levels and heat dissipation. It operates efficiently and quietly in a normal office environment and can be conveniently positioned beneath a desk or table.

▪ VAX-11/725 BASIC SYSTEM

The VAX-11/725 contains a VAX-11/730 CPU, up to 3 Mbytes of main memory, a dual, TU58 tape cartridge drive as the console load device, and the RC25 mass-storage disk drive subsystem. The RC25 subsystem uses Winchester technology and contains an intelligent controller, 26 Mbytes of data storage on an 8-inch fixed disk, and 26 Mbytes of disk storage on a removable cartridge. The RC25 contains onboard diagnostic tests to insure reliable operation and facilitate maintenance. The execution time of floating-point instructions and some integer instructions can be decreased by the addition of the FP730 floating-point accelerator option. Figure 1-2 shows the components included with the basic system.

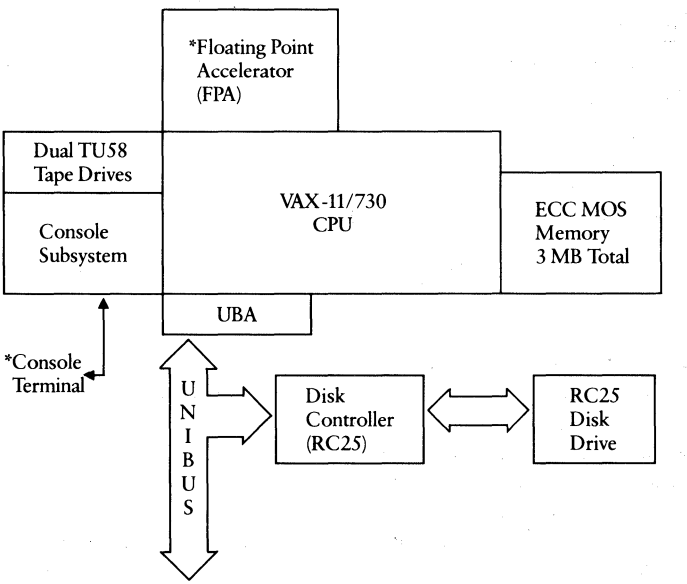


Figure 1-2 ▪ VAX-11/725 System Hardware Configuration

The console subsystem is a microprocessor-controlled, intelligent interface that communicates with the console terminal and with the remote diagnostic facility. The console terminal is a communications device for the user, system manager and service engineer; together with the console command language it provides a valuable tool for debugging programs and performing maintenance functions. Each console terminal has a standard keyboard. The VT100 or VT200 series of video display terminals or the LA100 or LA12 printers can be selected as the console device. Two TU58 cartridge tape drives communicate with the console subsystem and are used to load the system microcode and to operate microlevel and macrolevel diagnostic programs. The TU58 drives can be used to bootstrap load the operating system, to load files into physical memory, and to store files. The switches and indicators on the control panel are monitored and controlled by the console subsystem.

Many standard UNIBUS interfaces are either included with the system or available as options. The UNIBUS cannot be expanded outside the system cabinet. The interfaces include the DMF32 multifunction communication controller and the DZ11 asynchronous serial-line interfaces. The DMR11 option allows synchronous high-speed communication with remote systems and terminals through common-carrier telephone lines.

The DEUNA is a high-performance synchronous communication controller used in local area network (LAN) applications. The DEUNA connects to the UNIBUS and allows Digital systems and terminals and systems developed by other manufacturers to communicate through the Ethernet network. Ethernet permits data transfer rates of up to 10 Mbits per second between processors or between processors and devices.

The VS100 adapter is an option that provides high performance and high-resolution text and graphic features in a workstations environment.

- VAX-11/725 PACKAGE SYSTEM CONFIGURATION

The VAX-11/725 is available in the following three systems configurations to meet specific customer applications. Each system contains the VAX-11/730 CPU, ECC MOS memory, an RC25 controller, and two TU58 cartridge tape drives as standard components.

- Package A is an entry level system that includes the VAX-11/730 CPU and 1 Mbyte of ECC MOS memory. One slot is dedicated to memory expansion and one slot is available for memory expansion or for a UNIBUS interface option. The remaining four slots are available for UNIBUS interface options. Six panel units locations at the rear of the cabinet are available for external cable connectors.

- Package B includes the VAX-11/730 CPU, 2 Mbytes of ECC MOS memory, and a DMF32 multifunction communication controller. One slot is available for memory expansion or for a UNIBUS option, and three backplane slots are available for UNIBUS interface options. Two connector panel units at the rear of the cabinet are available for external cable connectors.

- Package C contains the VAX-11/730 CPU, 2 Mbytes of ECC MOS memory, an FP730 floating-point accelerator, a DMF32 multifunction controller, and a DEUNA Ethernet controller. One backplane slot is available for memory or UNIBUS expansion. One panel unit location at the rear of the cabinet is available for an external cable connector.

VAX-11/730 Low-end UNIBUS System

The VAX-11/730 processor system is an expandable low-end member of the VAX family that can extend VAX capabilities to almost any environment. It provides up to 24 users with the large program capacity and high-performance features of the larger VAX systems while incorporating bit-slice and programmed array logic for low power consumption and efficient operation. Space is included in the main cabinet for a variety of device interfaces and communication options. The VAX-11/730 system is contained in a low-height, single-width cabinet that can be placed within an existing office or laboratory space. As a result, the VAX versatility and performance can be located at a project or section level within a company.

- VAX-11/730 BASIC SYSTEM

The basic VAX-11/730 computer system contains the VAX-11/730 CPU, up to 3 Mbytes of ECC MOS memory, and two TU58 cartridge tape drives as the console load devices. When I/O device expansion is required, the internal UNIBUS expansion backplane can be used, or a UNIBUS expansion cabinet the same size as the CPU cabinet can be added externally to the system. To increase the mass storage capabilities of the VAX-11/730 system, several disk and tape drive controllers are available for addition to the system cabinet or for mounting in a UNIBUS expansion cabinet. The UDA50 universal disk adapter can be added to support a combination of high-capacity disk drives. The execution time of floating-point instructions and some integer instructions can be decreased by the FP730 floating-point accelerator option, which mounts within the system cabinet. The basic system components are shown in figure 1-3.

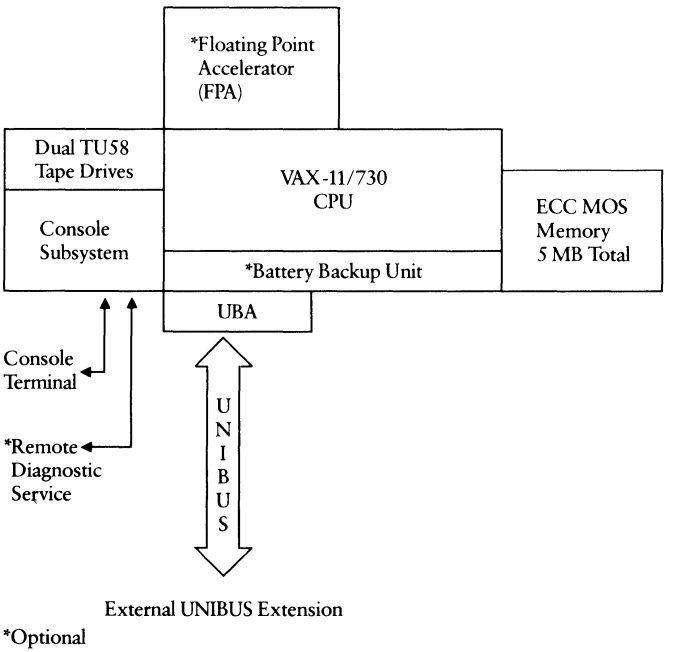


Figure 1-3 ▪ VAX-11/730 SBB System Hardware Configuration

The console subsystem is a microprocessor-controlled, intelligent interface that communicates with the console terminal and the remote diagnostic facility. The console terminal is a communications device for the user, system manager, or service engineer and together with the console command language it provides a valuable tool for debugging programs and performing maintenance functions. The console terminals available with the VAX-11/730 are the LA120 freestanding printer, and the LA100 and LA12 tabletop printers. The printers include a standard keyboard. The remote diagnostic port enables diagnostic maintenance to be performed on the system from a host processor located at a remote Digital facility. The remote diagnostic service is supplied with a Digital Field Service contract.

Two TU58 cartridge tape drives communicate with the console subsystem and are used to load the system microcode and to perform microlevel and macrolevel diagnositic programs. The TU58 drives can be used to bootstrap load the operating system, to load files into physical memory, and to store files that describe and execute bootstrap procedures that are used on a specific system. The switches and indicators on the control panel operate through the console subsystem and are used to monitor and control the operation of the system.

Many standard communication interfaces can be installed on the UNIBUS, including the DMF32 multifunction communication controller and the DHU11, DMZ32, and DZ11 asynchronous serial-line interfaces. The DMP11, DMR11, and DUP11 options also can be included to allow synchronous high-speed communication with remote systems and terminals through common-carrier telephone lines. Up to 24 asynchronous communication lines can be connected to the basic system to provide local or remote terminal access.

The DEUNA is a high-performance synchronous communication controller used in local area network (LAN) applications. The DEUNA connects to the UNIBUS and allows communication through Ethernet with Digital systems and terminals and systems developed by other manufacturers. It permits data transfer rates of up to 10 Mbits per second between processors or between processors and devices.

The LP25 and the LP26 lineprinters are available to provide hardcopy output from the system. The LP25 prints at 300 lines per minute and the LP26 at 600 lines per minute. The printers connect to the system through the DMF32 printer port or through a separate controller that mounts on the UNIBUS.

The UDA50 universal disk adapter can be installed on the UNIBUS and allows control of up to four disk drive devices, including the RA80 (121-Mbyte) and RA81 (456-Mbyte) fixed disk drives, and the RA60 (205-Mbyte) removable disk drive in any combination. The UDA50 is an intelligent controller designed for use with single-CPU systems operating within the Digital Storage Architecture.

Up to four RL02 (10.4-Mbyte) disk drives, the RC25 (52 Mbyte) fixed- and removable-disk drive and the TU80 and TU81 magnetic tape drives are available to operate with the VAX-11/730 processor.

The H9642 general purpose UNIBUS expansion cabinet provides space to mount the BA11-K expander box into which the DD11 UNIBUS backplane can be installed.

The H7750 memory battery backup unit can be included with the VAX-11/730 processor to maintain the integrity of the data in memory in the event of a power failure.

- VAX-11/730 SYSTEM BUILDING BLOCK CONFIGURATION

The VAX-11/730 processor is provided in several VMS and ULTRIX-32 operating system packages that can be configured for specific customer applications. The system is available with either a VMS operating software license and warranty or a license for up to 16 users of the ULTRIX-32 operating software. The basic packaged system includes the VAX-11/730 CPU, 1 Mbyte of ECC MOS memory, an integrated disk controller, and the DMF32 multifunction communications controller for which a variety of add-on options are available. The VAX-11/730 VMS and ULTRIX-32 systems are packaged with an R80 (121-Mbyte) fixed-disk drive and an RL02 (10.4-Mbyte) removable-cartridge disk drive mounted in the CPU cabinet, or with a R80 disk drive and the TS05 magnetic tape transport mounted in the CPU cabinet.

- VAX-11/730 PACKAGED SYSTEM CONFIGURATION

The VAX-11/730 is available in the following package system configurations that operate with both VMS and ULTRIX-32 software. Each system includes the VAX-11/730 CPU, 1 Mbyte of memory, and a mass storage subsystem. An I/O connection panel at the rear of the cabinet contains the cable connector for the console terminal and space for mounting additional connector panel units.

- The VAX-11/730 R80/TSU05 system includes one R80 fixed-disk drive system device connected to the RB730 integrated disk controller, one TSU08 (48-Mbyte) streaming magnetic tape drive used as a backup load device, and a DMF32 multifunction communications controller. The system is contained in a single cabinet and includes an LA100 console printer terminal. The CPU backplane has prewired designated slots for an additional 3 Mbytes of main memory and for an FP730 floating-point accelerator module.

- VAX-11/730 R80/RL02 system includes the R80 fixed disk system device, an RL02 (10.4-Mbyte) removable-disk drive as a backup load device, and the DMF32 controller. Both disk drives connect to the RB730 integrated disk controller. The system is contained in a single cabinet and includes an LA12 or LA100 console terminal. The CPU backplane has prewired designated slots for four additional Mbytes of memory and for the FP730 module.

VAX-11/750 Midrange System

The VAX-11/750 processor system is the midrange member of the VAX family and incorporates many innovations designed to increase performance and to reduce the overall cost to owners. The VAX-11/750 system supports up to 64 users and is the smallest VAX member that can be part of the VAXcluster architecture. It also operates with Digital's intelligent disk controllers to offload the disk management functions from the host processor. The VAX-11/750 is enclosed in a low-height, extended-width cabinet and implements custom gate-array technology. The gate arrays are bipolar LSI Schottky logic designed by Digital specifically for this system to reduce the number of components to lower power consumption.

- VAX-11/750 BASIC SYSTEM

  The basic VAX-11/750 system configuration is shown in figure 1-4. It contains the VAX-11/750 CPU, up to 8 Mbytes of ECC MOS memory, and a TU58 cartridge tape drive as the console load device. A 4-Kbyte bipolar cache memory with parity is included to improve the processor performance. A 1-Kbyte user writable control store (WCS) option provides customers with the capability of executing application-specific microroutines. The execution time of floating-point instructions and some integer instructions can be decreased by adding the FP750 floating-point accelerator option, which mounts within the system cabinet. In addition, the KU750 extended-range G and H floating-point data type option can be added to further enhance floating-point operations. When I/O device expansion is required, the internal UNIBUS expansion backplane can be used or a low-height, single-width UNIBUS expansion cabinet can be added.

  To increase the mass storage capabilities of the VAX-11/750 system, several disk and tape drive controllers are available and can be included in the system cabinet or in the UNIBUS expansion cabinet. The UDA50 universal disk adapter supports a combination of high-capacity disk drives.

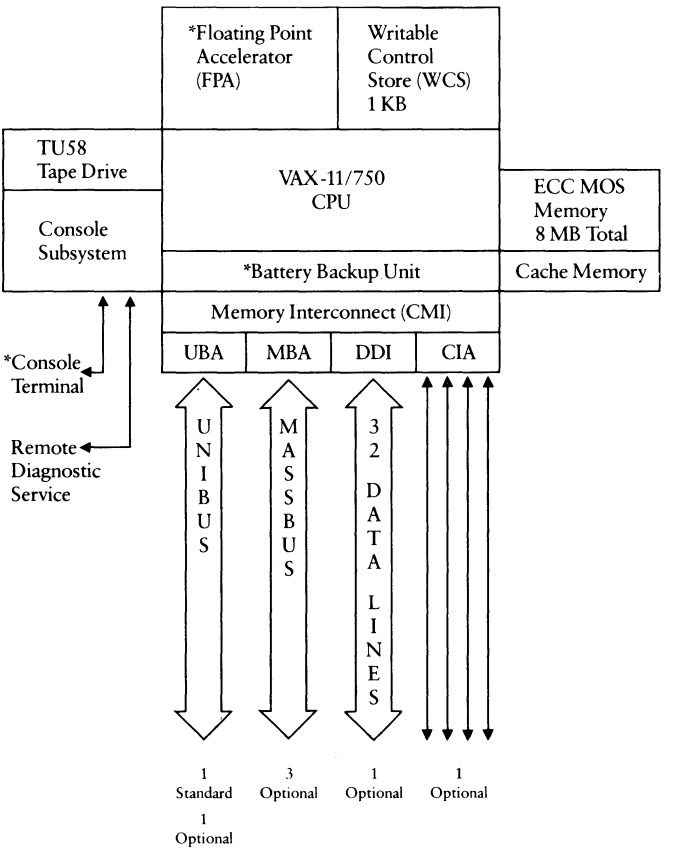


Figure 1-4 • VAX-11/750 System Hardware Configuration

The console subsystem is a microprocessor-controlled, intelligent interface that communicates with the console terminal and with the remote diagnostic facility. The console terminal is a communications device for the user, system manager, or service engineer and together with the console command language it provides a valuable tool for debugging programs and performing maintenance functions. Console terminals that are available for the VAX-11/750 are the LA120, LA100, and LA12 printers, each with a standard keyboard. The remote diagnostic service is an option that allows diagnostic programs to be performed on the system from a host processor at a remote Digital facility. The remote diagnostic service is obtained under a Digital Field Service contract. The TU58 cartridge tape drive, which is part of the console subsystem, is used to load the system microcode and to perform microlevel and macrolevel diagnositic programs. The TU58 is also used to bootstrap load the operating system, to load files into physical memory, to update software, and to store files that describe and execute bootstrap procedures used with a specific system. The switches and indicators on the control panel operate through the console subsystem and are used to monitor and control the operation of the system.

A DW750 UNIBUS adapter (UBA) is included with the processor and a second UBA can be added to the system. Up to three RH750 MASSBUS adapter (MBA) options can be added, provided only one UBA is installed, or two MBA options can be added when two UBA options are installed. The MBA provides fast data access to the mass storage devices.

Many standard communication interfaces can be installed on the UNIBUS, including the DMF32 multifunction communication controller and the DHU11, DMZ32, and DZ11 asynchronous serial-line interfaces. The DMP11, DMR11, and DUP11 options can also be included to allow synchronous high-speed communication with remote systems and terminals through common-carrier telephone lines. Up to 24 asynchronous communication lines can be implemented to the basic system to provide local or remote terminal access.

The DEUNA is a high-performance, synchronous communication controller used in local area network applications. The DEUNA connects to the UNIBUS and allows communication through Ethernet with Digital systems and terminals and systems developed by other manufacturers. It permits data transfer rates of up to 10 Mbits per second between processors or between processors and devices.

A maximum of four high-speed printers can be installed, including LP25 and LP26 lineprinters. These printers provide hardcopy output: the LP25 prints at 300 lines per minute and the LP26 at 600 lines per minute. The printers connect to the system through the DMF32 printer port or through a separate controller that mounts on the UNIBUS.

The UDA50 universal disk adapter can be installed on the first UNIBUS and two UDA50 options can be installed on the second UNIBUS. Each UDA50 controls up to four disk drive devices in any combination, including the RA80 (121-Mbyte) and the RA81 (456-Mbyte) fixed-disk drives, and the RA60 (205-Mbyte) removable-disk drive. The UDA50 is an intelligent controller designed for use with single-CPU systems operating within the Digital Storage Architecture.

The UNIBUS supports up to four RL02 (10.4-Mbyte) removable-disk drives, one RC25 (52-Mbyte) fixed- and removable-disk drive, and two TU80 or two TU81 magnetic tape drives.

A total of eight disk drives or eight magnetic tape formatters, in any combination, can be connected to each MBA. The devices include the RM05 (256-Mbyte) removable-disk drives, and the RP07 (516-Mbyte) fixed-disk drive, which is supported only as a data storage. The magnetic tape drives include the TE16, TU77, and TU78 and units.

The CI750 Computer Interconnect adapter (CIA) is a microprocessor-controlled, high-speed interface between the memory interconnect of the CPU and the dual-path CI bus. It allows information to be transferred at 70 Mbits per second between the VAX-11/750 and the VAXcluster components. The CI750 adapter connects to the SC008 Star Coupler through four coaxial cables with a maximum length of 45 meters (147.6 feet).

The DR32 device interconnect (DDI) is a high-performance general purpose interface that enables communication between VAX processors and between VAX processors and user devices. It connects to the internal CPU/memory-interconnect (CMI) bus and provides a 32-bit parallel data path and an 8-bit parallel control path. The DDI transfers blocks of sequential data to and from memory at rates up to 6.67 Mbytes per second through direct memory access (DMA) transfers.

The H7112 memory battery backup unit is available to maintain the integrity of the data in memory in the event of a power failure.

The H9642 general purpose UNIBUS expansion cabinet provides space to mount the BA11-K expander box in which the DD11 UNIBUS backplane can be installed.

- VAX-11/750 SYSTEM BUILDING BLOCK CONFIGURATION

The VAX-11/750 processor is provided in two basic system configurations—one that includes the VAX-11/750 CPU with VAX/VMS operating software license and one with the VAX-11/750 CPU and an ULTRIX-32 operating software license for up to 16 users. The mass storage devices, communication interfaces, and console devices options are selected from the System Building Block Menus. The operating software media and documentation are selected from the menu according to the mass storage media.

Each basic system contains the VAX-11/750 processor and 2 Mbytes of ECC MOS main memory. The total memory can be expanded to 8 Mbytes. An H9642 general purpose UNIBUS expansion cabinet is available with the basic system to allow expansion of the system I/O capabilities. The mass storage devices include system and load devices that are selected from the menu to conform to the system mass storage requirements. The system storage devices include the RA60, RA80, RA81, and RM05 disk drives. The system load devices are the TU77, TU78, TU80, and TU81 magnetic tape units and the RA60 disk drive. The communication interfaces and controllers are the DMF32, DZ11, DHU11, DMZ32, and the DEUNA options.

- VAX-11/750 VAXCLUSTER CONFIGURATIONS

The VAX-11/750 VAX/VMS systems are available in a VAXcluster system building block (SBB) configuration and a VAXcluster upgrade configuration. These systems are designed to operate within the VAXcluster architecture. Each system includes the VAX-11/750 CPU, 4 Mbytes of ECC MOS memory, a CI750 adapter and connecting cables, the VAX/VMS operating system license and warranty, and the DECnet software license.

The VAX-11/750 VAXcluster SBB configuration contains the VAXcluster hardware, including the SC008 Star Coupler unit and an HSC50 intelligent mass-storage server unit, to which other VAX systems and mass storage devices can be connected. The system storage devices that can be selected from the menu include RA60, RA80, and RA81 disk drives. The communication interfaces and controllers on the menu are the DMF32, DZ11, DHU11, DMZ32, and the DEUNA options.

The VAXcluster upgrade configuration is designed to attach to the existing VAXcluster architecture through the CI750 adapter and cables. The system storage devices that can be selected by menu are the RA60, RA80, and RA81 disk drives. The communication interfaces and controllers on the menu are the DMF32, DZ11, DHU11, DMZ32, and the DEUNA options.

VAX-11/780 Original VAX System

The VAX-11/780 computer system is the original VAX system developed to meet the needs of users with large databases and extensive processing requirements. This high-performance system uses proven Schottky TTL (transistor-transistor logic) technology and can support more than 100 active users. The processor is contained in a full-height, double-width cabinet and provides space for device interfaces and controllers and for communication options. The VAX-11/780 system supports a complete line of peripheral devices and is compatible with Digital Network Architecture, Digital Storage Architecture, and VAXcluster architecture.

- VAX-11/780 BASIC SYSTEM CONFIGURATION

The basic VAX-11/780 system is shown in figure 1-5 and contains the VAX-11/780 CPU, up to 16 Mbytes of ECC MOS main memory, an RX01 console load device and I/O expansion capabilities. The H9652 full-height, single-width, UNIBUS expansion cabinet also is included with the system. An additional 16 Mbytes of ECC MOS memory can be added to the system by including the H9652-H optional CPU expander cabinet. The execution time of floating-point instructions and some integer instructions can be decreased by the addition of the FP750 floating-point accelerator option in the CPU backplane. The KU780 option is a 2-Kbyte user-control-store option that also mounts in a dedicated slot of the CPU backplane and provides control store space for the KE780 extended-range G and H floating-point data type option and the QE109-CY microprogramming tools option.

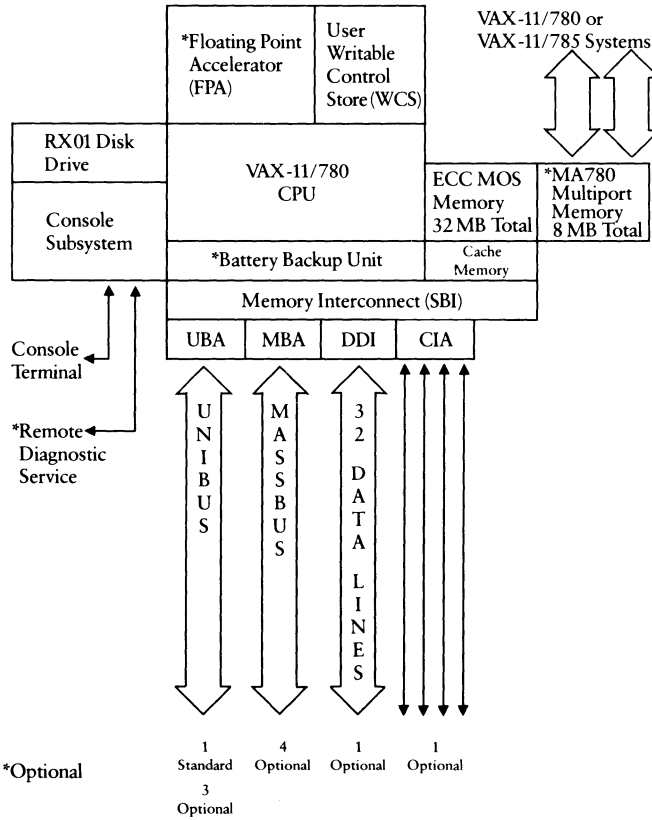


Figure 1-5 • VAX-11/780 System Hardware Configuration

The console subsystem is a microprocessor-controlled, intelligent interface that communicates with the console terminal and with the remote diagnostic facility. The console terminal is a communication device for the user, system manager, or service engineer. Together with the console command language it provides a valuable tool for debugging programs and performing maintenance functions. The console terminals available for the VAX-11/780 system are the LA120 freestanding printer and the LA100 tabletop printer. The terminals include a standard keyboard. The remote diagnostic service is an option that allows diagnostic programs to be performed on the system from a host processor at a Digital facility. The remote diagnostic service is obtained by Digital Field Service contract. The RX01 disk drive is part of the console subsystem and is used to load the system microcode and to perform microlevel and macrolevel diagnostic programs. The RX01 disk drive also is used to bootstrap load the operating system, to load files into physical memory, to update software, and to store files that describe a specific system and execute bootstrap procedures that are used on a system. The switches and indicators on the control panel operate through the console subsystem and are used to monitor and control the operation of the system.

A DW780 UNIBUS adapter (UBA) is included with the processor and three UBA options can be added to the system. A total of four MASSBUS adapter (MBA) options can be included when only one UBA is installed. When two UNIBUS adapters are installed, two MBA options can be included. The MBA provides fast access to the data on the mass storage devices.

Many standard communication interfaces can be installed on the UNIBUS, including the DMF32 multifunction communication controller and the DHU11, DMZ32, and DZ11 asynchronous serial-line interfaces. The DMP11, DMR11, and DUP11 options can also be included to allow synchronous high-speed communication with remote systems and terminals through common-carrier telephone lines. Up to 24 asynchronous communication lines can be connected to the basic system to provide local or remote terminal access.

The DEUNA is a high-performance synchronous communication controller used in local area network (LAN) applications. The DEUNA connects to the UNIBUS and allows communication through the Ethernet network with Digital systems and with terminals and systems developed by other manufacturers. The DEUNA permits data transfer rates of up to 10 Mbits per second between processors or between processors and devices.

A UDA50 universal disk adapter (UDA) can be installed on the first UNIBUS and two UDA50 options can be installed on each of the three UNIBUS options. Each UDA50 can control up to four disk drive devices, including the RA80 (121-Mbyte) and the RA81 (456-Mbyte) fixed-disk drives, and the RA60 (205-Mbyte) removable-disk drive in any combination. The UDA50 is an intelligent controller designed for use with single CPU systems that operate within the Digital Storage Architecture.

A maximum of 16 high-speed line printers can installed, including LP25 and LP26 lineprinters. These printers provide hardcopy output: the LP25 prints at 300 lines per minute and the LP26 prints at 600 lines per minute. The printers connect to the system through the DMF32 printer port or through a separate controller that mounts on the UNIBUS.

The UNIBUS also supports up to four RL02 (10.4-Mbyte) removable-disk drives, one RC25 (52-Mbyte) fixed- and removable-disk drive, and two TU80 or two TU81 magnetic tape drives.

Disk drives or magnetic tape formatters in any combination totaling eight can be connected to each MBA. The devices include the RM05 (256-Mbyte) removable-disk drives, and the RP07 (516-Mbyte) fixed-disk drive, which is supported only as a data storage. The magnetic tape drives include the TE16, TU77, and TU78 unit.

The CI780 Computer Interconnect adapter (CIA) is a microprocessor-controlled, high-speed interface between the synchronous backplane interconnect (SBI) of the CPU and the dual-path CI bus. It allows information to be transferred at 70 Mbits per second between the VAX-11/780 and the VAXcluster components. The CI750 adapter connects to the SC008 Star Coupler through four coaxial cables with a maximum length of 45 meters (147.6 feet). Through the VAXcluster network, the overall system efficiency is greatly increased by sharing the processor loads and by providing access to common mass storage facilities.

The DR32 device interconnect (DDI) is a high performance general-purpose interface that permits communication between VAX processors and between VAX processors and user devices. It connects to the SBI bus and provides a 32-bit parallel data path and an 8-bit parallel control path. The DDI transfers blocks of sequential data to and from memory at rates up to 6.67 Mbytes per second through direct memory access (DMA) transfers.

One H7112 memory battery backup unit is required for each memory controller on the VAX-11/780 VAX/VMS system to maintain the integrity of the data in memory in the event of a power failure.

- VAX-11/780 SYSTEM BUILDING BLOCK CONFIGURATIONS

  The VAX-11/780 processor is provided in two basic system configurations—one that includes the VAX-11/780 CPU and a VAX/VMS operating software license and warranty, and one with a VAX-11/780 CPU and an ULTRIX-32 software license for up to 16 users. Mass storage devices, communication interfaces, and console devices options are selected from the System Building Block Menus. The operating software media and documentation are selected from the menu according the type of mass storage media selected.

  Each basic system contains the VAX-11/780 processor, 2 Mbytes of ECC MOS main memory, and an H9652 UNIBUS expansion cabinet. The total memory can be expanded to a maximum of 8 Mbytes. The mass storage devices include a system device and load devices that are selected from the menu to conform to the mass storage requirements. The system storage devices include the RA60, RA80, RA81, and RM05 disk drives. The system load devices include the TU77, TU78, TU80, and TU81 magnetic tape units and RA60 disk drive. The communication interfaces and controllers are the DMF32, DZ11, DHU11, DMZ32, and the DEUNA options.

- VAX-11/780 VAXCLUSTER CONFIGURATION

  The VAX-11/780 VMS systems are available in a VAXcluster system building block (SBB) configuration and a VAXcluster upgrade configuration. These systems are designed to operate within the VAXcluster architecture. Each system includes the VAX-11/780 CPU, 4 Mbytes of ECC MOS memory, a CI750 adapter and connecting cables, an H9652 UNIBUS expansion cabinet, a VAX/VMS operating system license and warranty, and a DECnet software license.

  The VAX-11/780 VAXcluster SBB configuration contains the VAXcluster hardware, including the SC008 Star Coupler unit and the HSC50 intelligent mass-storage server unit, to which other VAX systems and mass storage devices can be connected. System storage devices that can be selected from the menu include the RA60, RA80, and RA81 disk drives. The load devices are the TU78 magnetic tape unit and RA60 disk drive. The communication interfaces and controllers on the menu are the DMF32, DZ11, DHU11, DMZ32, and the DEUNA options.

  The VAXcluster upgrade configuration is designed to attach to an existing VAXcluster configuration through the CI750 adapter and cables. System devices that can be selected from the System Building Block Menu are the RA60, RA80, and RA81 disk drive. The communication interfaces and controllers on the menu are the DMF32, DZ11, DHU11, DMZ32 and the DEUNA options.

VAX-11/782 Attached-processor System

The VAX-11/782 attached-processor system is a tightly coupled, asymmetrical multiprocessor system based on a MA780 multiport shared memory. The system can provide up to 1.8 times the performance of the VAX-11/780 system. The VAX-11/782 consists of two VAX-11/780 processors and a 4-Mbyte MA780 shared main memory mounted between the VAX CPU cabinets. The main memory can be expanded to 8 Mbytes by adding a second MA780 multiport memory cabinet containing 4 Mbytes of memory. One VAX-11/780 CPU, designated the primary processor, contains 256 Kbytes of local ECC MOS memory for diagnostics programs, a shared memory interface, and a DW780 UNIBUS adapter. All I/O devices and peripheral devices connect to the primary processor. The second CPU is designated the attached processor and contains 256 Kbytes of local memory for diagnostic testing and a shared memory interface. The attached processor does not support I/O devices.

The UNIBUS and MASSBUS adapters and the CI and DDI adapters available for the primary processor are the same as those used with the single VAX-11/780 processor system. The mass storage devices available for the primary processor are also the same as those used with the single VAX-11/780 processor system. The VAX-11/782 processor system cannot be connected to VAXcluster components. A UNIBUS expansion cabinet can be added to the VAX-11/782 system. The FP782 floating-point accelerator option can be added to each VAX-11/780 processor to decrease the execution time of the floating-point operations and some of the integer arithmetic operations. Figure 1-6 shows the hardware configuration of the VAX-11/782 system.

- VAX-11/782 SYSTEM BUILDING BLOCK CONFIGURATION

The VAX-11/782 attached processor system operates with VMS operating software to provide a reliable high-performance environment for the shared processor operations. The basic system includes two VAX-11/780 processors, 4 Mbytes of ECC MOS shared memory, an H9652 UNIBUS expansion cabinet with a BA11-K expansion box, and DD11-DK backplane. The total shared memory can be expanded to 8 Mbytes. Mass storage, communication, and console devices and interface options can be added to the basic system from the System Building Block Menus. They are the same as those listed on the VAX-11/780 VAX/VMS System Building Block Menu.

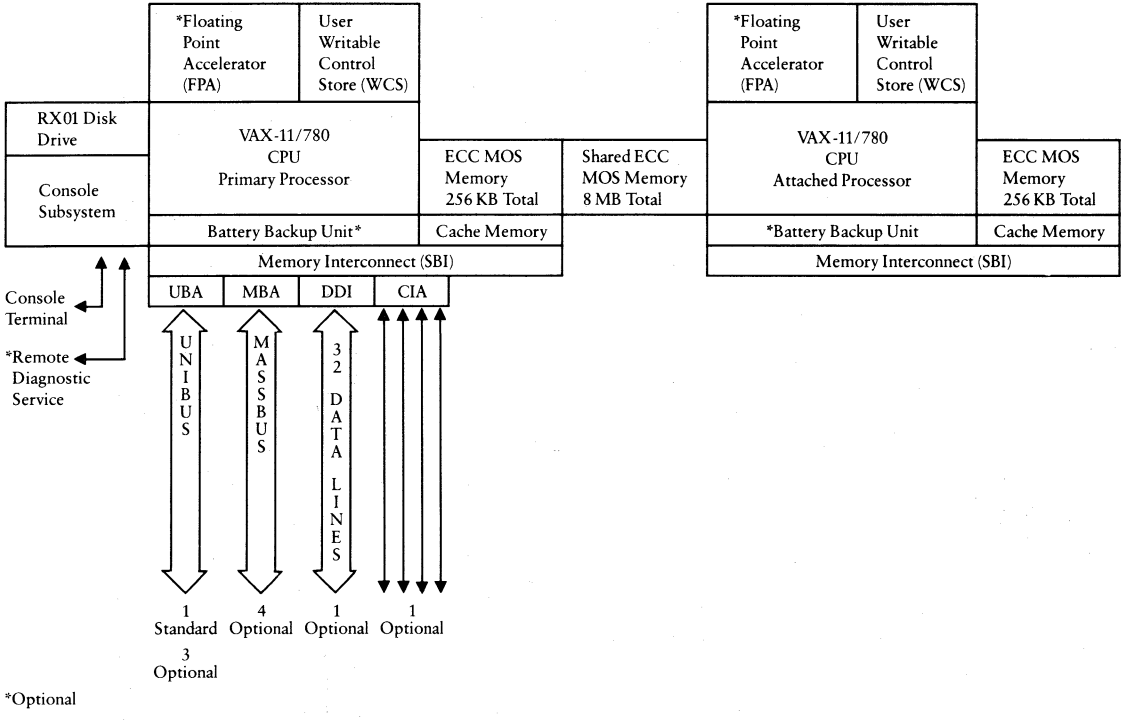


Figure 1-6 ▪ VAX-11/782 Attached-processor System Hardware Configuration

The operating software media and documentation is also selected from the menu according to the mass storage devices selected. The UNIBUS expansion cabinet is included with the basic system to allow expansion of the system I/O capabilities. The mass storage devices include both a system device and load devices that are selected from the menu to conform to the system data storage requirements. The system storage devices include the RA60, RA80, RA81, RM05, and RP07 disk drives and the system load devices include the TU77, TU78, TU80, TU81 magnetic tape units and the RA60 disk drive. The communication interfaces and controllers are the DMF32, DZ11, DHU11, DMZ32, and DEUNA options.

VAX-11/785 Extended-performance System

The VAX-11/785 extended-performance system was designed with Fairchild Advanced Schottky Technology FAST* and improved logic circuits to provide up to 1.5 times the performance of the VAX-11/780 processor. The system is available in the same basic system configurations as the VAX-11/780 processor shown in figure 1-5. It includes VMS and ULTRIX-32 operating systems and the VAXcluster and VAXcluster upgrade configurations. The VAX-11/785 processor contains the standard instructions for packed-decimal floating- (G and H data types) and fixed-point arithmetic, character, and string manipulations. It also includes RAM-based microcode, 8 Kbytes of writable-control-store, 48-Kbyte console memory, and a 32-Kbyte two-way set-associative cache memory. The FP785 floating-point accelerator option can be added to the VAX-11/785 processor to decrease the execution time of the floating-point operations and some of the integer arithmetic operations. The system is also available as a replacement CPU module set; for a small increase in cost, the improved CPU logic modules can be installed in an existing VAX-11/780 or VAX-11/782 system to provide more processing power, higher throughput, and the ability to support more users. All other system hardware and software configurations are the same as described the VAX-11/780 system.

*FAST is a trademark of Fairchild Camera and Instrument Corporation.

VAX 8600 High-performance System

The VAX 8600 processor is the most powerful member of the VAX family and is the first VAX system designed for large-scale VAX computing. The VAX 8600 processor provides up to 4.2 times the performance of the industry standard VAX-11/780 processor and has the power, performance and capacity required for the high-speed processing of large applications. It conforms to the VAX architecture, making it compatible with other VAX processors.

The VAX 8600 system supports extensive scientific, engineering, and realtime applications, including artificial intelligence, simulation, and computer-aided design. It also supports large, general purpose timesharing applications in government, commercial, administrative, and academic environments. It provides the performance required for large, complex multiuser applications and the I/O capacity for large timesharing applications. It provides VAX customers with a means of expanding PDP-11, VAX-11/725, VAX-11/730, VAX-11/750, VAX-11/780, VAX-11/782, VAX-11/785, DECsystem-10, and DECSYSTEM-20 systems to meet new application requirements.

The VAX 8600 processor incorporates the latest in proven technological advancements, including customized emitter-coupled (ECL) gate-array logic and four-stage instruction pipeline processing. The customized ECL logic allows more components to be included on a module and provides a significant increase in processing speed. The pipeline processing of instructions enables the processor to operate on four instructions simultaneously, thereby reducing the number of cycles required for each instruction. A 16-Kbyte write-back cache memory maintains the processing speed by updating the main memory during memory access operations.

The VAX 8600 processor includes virtual memory management, a bootstrap loader, standard instructions for packed decimal, floating (F, D, G, and H data types) and fixed-point arithmetic, and character and string manipulations. Also included is a 16-Kbyte write-back cache memory, high-precision realtime programmable clock, time-of-year clock with battery backup, and 8 Kwords (86-bit words) of writable control store.

The VAX 8600 processor includes a synchronous backplane interconnect adapter (SBIA) and a second adapter SBIA can be added to expand the I/O capabilities of the system. The communications path between the CPU and main memory is through internal memory bus, which effectively decreases the memory access time. The SBIA is used only for communications between devices and memory; therefore, the speed and efficiency of the I/O transfers are increased.

The dependable performance of large program applications is ensured by extensive self-diagnostic and maintenance features incorporated in the VAX 8600 system. The automatic error checking and correcting of main memory and cache memory data eliminates many types of errors without interrupting the system operation. When a problem exists, an internal diagnostic bus, connected to all main logical elements, can be used to sample data and isolate the failing logic.

Environmental sensors monitor the physical environment of the VAX 8600 system and provide a warning when the environmental conditions are not acceptable.

- VAX 8600 BASIC SYSTEM

The basic configuration of the VAX 8600 processor is shown in figure 1-7. Up to 32 Mbytes of ECC MOS main memory can be included in each processor. Each memory array contains 4 Mbytes of storage implemented with 256-Kbit integrated circuits (ICs). The FP86 floating-point accelerator (FPA) is included with the processor and operates with the CPU to substantially decrease the instruction execution time of all floating-point arithmetic instructions, floating-point data types, and some of the integer arithmetic operations.

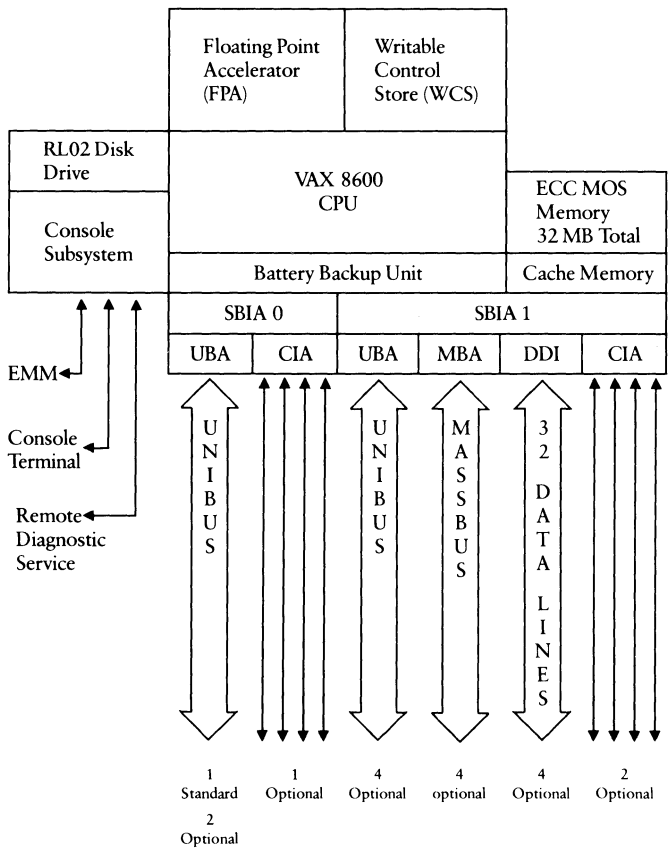


Figure 1-7 • VAX 8600 System Hardware Configuration

The console subsystem is an intelligent microprocessor-controlled interface for the console terminal, console load device, environmental monitoring module circuits, remote diagnostic port, and control panel. The LA100 tabletop printer is used as the console terminal and includes a standard keyboard. The console load device is an RL02 disk drive providing 10.4 Mbytes of storage on a removable disk. The RL02 drive is used to load the system microcode, to perform microlevel and macrolevel diagnostic programs, to bootstrap load the operating system, to load files into physical memory, and to store files that describe and execute bootstrap procedures that are used with a specific system. The environmental monitoring module checks and controls the internal temperature, cooling air flow, power supply voltages and other conditions within the main system cabinets. The remote diagnostic port connects to a Digital remote diagnostic service center, which can perform diagnosic tests on the VAX 8600 system. A DF112 modem is supplied for remote port communications with systems installed in the United States and Canada. The switches and indicators on the control panel are monitored and controlled by the console subsystem.

The I/O subsystems connect to the CPU through the DB86 synchronous backplane interconnect adapters (SBIA 0 and SBIA 1). SBIA 0 is included with the system and SBIA 1 can be added in the CPU cabinet. The SBIA 0 bus can be extended from the CPU cabinet into an SBI expansion cabinet. A DW780 UNIBUS adapter (UBA) is included with the system and connects to SBIA 0, and two UBA options can be added to SBIA 0. The CI780 computer interconnect adapter (CIA) is optional and can also be connected to SBIA 0 for VAXcluster configuration.

UNIBUS, MASSBUS, CI, and DDI adapters can be installed on SBIA 1. A maximum of four UBA options, four optional RH780 MASSBUS adapters (MBAs), four optional DR780 DR32 adapters (DDIs), or two CIA options can be installed on SBIA 1. The combination of UBA, MBA, CIA, and DDI options on the SBIAs depends on the total number and type of adapters installed.

Several standard UNIBUS communication interfaces may be included with the system and many are available as options. These interfaces include the DMF32 multifunction communication controller and the DHU11, DMZ32, and DZ11 asynchronous serial-line interfaces. The DMP11, DMR11, DR11, and DUP11 options can also be included to allow synchronous high-speed communication with remote systems through common-carrier telephone lines.

The DEUNA is a high-performance synchronous communication controller used in local area network applications. The DEUNA connects to the UNIBUS and allows communication through the Ethernet network with Digital's systems and terminals, and systems developed by other manufacturers. The DEUNA permits data transfer rates of up to 10 Mbits per second between processors or between processors and devices.

One UDA50 universal disk controller can be installed on the UNIBUS to control up to four disk drives, including the RA60 (205-Mbyte) removable-disk drive, the RA80 (121-Mbyte) fixed-disk drive, and the RA81 (456-Mbyte) fixed-disk drive, in any combination. The UDA50 is an intelligent controller designed for use with single-CPU systems that operate within the Digital Storage Architecture.

A maximum of 16 high-speed line printers can be installed, including LP25, LP26, and LP27 lineprinters. These printers provide hardcopy output: the LP25 prints at 300 lines per minute, the LP26 prints at 600 lines per minute, and the LP27 at 1,200 lines per minute. The printers connect to the system through the DMF32 printer port or through a separate controller that mounts on the UNIBUS.

The UNIBUS also supports up to four RL02 (10.4-Mbyte) removable-disk drives and a maximum of four TU81 magnetic tape drives.

Disk drives and magnetic tape formatters in any combination up to a total of eight can be connected to each MBA. The devices include the RM05 (256-Mbyte) removable-disk drive and the TE16, TU77, and TU78 magnetic tape units.

The CI780 computer interconnect adapter (CIA) is a microprocessor-controlled, high-speed interface that connects the SBI bus to the dual-path CI bus. Information is transferred at 70 Mbits per second between the VAX 8600 and the VAXcluster components. The CI780 adapter cables connect to the SC008 Star Coupler unit through two receive and two transmit coaxial cables, each with a maximum length of 45 meters (147.6 feet). Through the VAXcluster, the overall system efficiency is greatly increased by sharing the processor loads and by providing access to common mass-storage facilities.

The DR780 DR32 device interconnect (DDI) is a high-performance, general purpose interface that permits communication between VAX processors and between VAX processors and user devices. It connects to the SBI bus and provides a 32-bit parallel data path and an 8-bit control path. The DDI transfers blocks of sequential data to and from memory at rates up to 6.67 Mbytes per second through direct memory access (DMA) transfers.

The H7112 battery backup unit is included with the VAX 8600 processor to maintain the integrity of up to 32 Mbytes of data in memory in the event of a power failure.

The H9652 UNIBUS expansion cabinet provides the space for mounting two BA11-A UNIBUS expansion boxes that contain the UNIBUS backplanes.

The H9652 SBI expansion cabinet provides space for mounting additional SBI interfaces, including DW780 UNIBUS adapters, RH780 MASSBUS adapters, CI780 adapters, and DR780 adapters.

- VAX 8600 VAXCLUSTER CONFIGURATION

The VAX 8600 systems are available in two VAXcluster configurations: the VAX 8600 VAXcluster system building block (SBB) and VAX 8600 VAXcluster upgrade. Each system includes the VAX 8600 CPU, 12 Mbytes of ECC MOS memory, the FP86 floating-point accelerator, the CI780 adapter and connecting cables, the DMF32 multifunction communication controller, four DMZ32-M asynchronous multiplexers, and the DEUNA Ethernet communications controller. The systems are supplied with a VAX/VMS operating software license and a DECnet-VAX full-function license. The main memory of each configuration can be expanded in 4-Mbyte increments to a total of 32 Mbytes.

The VAX 8600 VAXcluster SBB system includes all the VAXcluster components needed to connect the VAX 8600 to other VAX systems and HSC50 intelligent mass-storage servers in a VAXcluster configuration. The system includes an SC008 Star Coupler unit, an HSC50 intelligent I/O mass storage server unit, and connecting cables. The system storage devices available are the RA60, RA80, and the RA81 disk drives. The load device is the TA78 magnetic tape unit. The type of software media, software support documentation and the level of software service is selected from the menu. The LA100 is used as the console terminal.

The VAX 8600 VAXcluster upgrade system is designed to be connected to existing VAXcluster configuration and includes the CI780 adapter and required interconnecting cables. No system or load devices are offered from the SBB menu. The LA100 is used as the console terminal.

· VAX System Characteristics

The VAX system hardware and software are combined to ensure efficient and reliable system operation. The VAX hardware architecture and the VMS or ULTRIX-32 operating systems and layered software and languages provide the compatibility required between VAX systems. To give VAX systems their impressive performance, the architecture incorporates 32-bit addressing, up to four billion bytes of virtual memory, an address translation and prefetch instruction buffer, an optional floating-point accelerator, and the powerful VAX instruction set. All VAX systems provide an attractive cost-performance ratio that makes the systems ideal for applications tailored for specific tasks. The high computational ability and large program size mean that VAX systems can handle tough realtime applications. The VMS operating system provides extensive facilities for good batch performance, including job control, multistream, spooled input and output, operator control, conditional command branching, and accounting functions. A choice of options–additional physical memory, user-control-store, additional peripheral equipment interfaces and others–allow even greater flexibility in configuring systems to optimize performance for specific applications.

Ease of Use

VAX systems are user-oriented and designed for easy operation. The Digital Command Language (DCL) used by the VAX/VMS system is easy to learn and suitable for both interactive and batch environments. The software compatibility of VAX systems allows software developed for one VAX system to run on another VAX system without modification. Because VAX systems use the same instruction set, users can take full advantage of another VAX system's capabilities. VMS provides extensive system management facilities, giving system managers and operators the tools necessary to control the system configuration and the operations of system users for maximum efficiency. Extensive help commands are available and the systems include complete multiuser security. Except for the MicroVAX, the VAX family of processors also implement a PDP-11 compatibility mode, which recognizes most of the PDP-11 instructions. With few exceptions, this allows users to execute code written for the PDP-11. The VAX console subsystem also contributes to the ease of use with a separate console terminal and a carefully designed console command language that lets the users perform such operations as examines and deposits, or bootstrap load the system program using simple commands. The console terminal also provides a hardcopy of console transactions. Switches on the front panel of the CPU can be set to reboot the system automatically, without operator intervention, in the event of a power failure or interrupted operation.

VAX systems are designed to facilitate rapid, low-cost applications development. With the complete set of VMS and ULTRIX-32 development tools, file system features, optional information management products, and other software packages, applications are easier to develop and require less debugging time. Digital's extensive educational services are also available to train and assist users in exploiting the wide and varied capabilities of VAX systems.

Installation and Maintenance

The VAX systems are easily tuned and adapted, so additional peripherals and options can be interfaced at any time. The VAXcluster network capability insures that the system will never be outdated because of limited processing power and mass data storage requirements. Customers may choose a wide variety of peripherals and packaging options to configure a VAX system to suit their requirements, whether the site is an office, a laboratory, or an industrial setting. Once the system is installed, extensive reliability, availability, and maintainability features in both the hardware and the software ensure data integrity and increase system uptime. Features such as ECC (error correction code) memory, online error logging, and a complete range of online and stand-alone diagnostics verify system integrity and help ensure proper system operation. The remote diagnosis option allows VAX system to be directly linked to a Digital diagnostic center for diagnosis of hardware and software failures. For smaller VAX systems, the diagnostics can be performed by a system user to verify proper hardware operation and quickly isolate system failures at the subsystem or device level. The remote support option, using remote diagnosis technology, gives the Digital service engineer a further level of technical resources.

Long-term Investment

The features of the VAX systems hardware and software, combined with the VAXcluster architecture and complete series of VAX processors, devices, and options, assure Digital's commitment to the continued development and support of VAX computers. These systems can be tailored to individual application requirements and they can be easily expanded or reconfigured if those needs should change in the future, an important consideration for customers involved in long-term projects and implementations.

Architecture Overview

The VAX family architecture is characterized by a powerful and complete set of 304 instructions, a wide range of data types, an efficient set of addressing modes, complete demand-paging memory management, and a large virtual address space of over 4 billion bytes.

- VAX INSTRUCTIONS AND DATA TYPES

VAX instructions and data vary in length and may begin at odd or even byte addresses without being aligned on longword boundaries in physical memory. Instructions not requiring arguments use only one byte, and other instructions may require up to 54 bytes, depending on the number of arguments and their addressing modes. The advantage of byte alignment is that instruction streams and data structures can be stored in less physical memory. Because the VAX instruction set is so flexible, most functions require fewer instructions and less storage than on other processors. The result is more compact and efficient programs, faster program execution, faster context switching, more precise and faster math functions, and improved compiler generated code.

The VAX native instruction set is an extension of the PDP-11 instruction set and can be grouped into classes based on their functions and uses:

- *Arithmetic and Logical Data Types* – These include integer, floating point, packed decimal, character string, and bit field. The data type identifies the number of stored bits to be considered as a unit and the method of interpreting the unit. Integer, floating point, packed decimal, and character data are stored starting on an arbitrary byte boundary. Bit and bit field data start on an arbitrary bit boundary. A collection of data structures can be packed together to use less storage space. The following data types may be used.
  - Integer are byte (8 bits), word (16 bits), longword (32 bits), and quadword (64 bits).
  - Floating point are 4-byte F_floating, 8-byte D_floating, 8-byte G_floating, and 16-byte H_floating.
  - Packed decimal are strings of up to 31 decimal digit bytes with two digits per byte.
  - Character strings are up to 64 Kbytes interpreted as a string of character codes. A numeric string is a character string of codes for decimal numbers up to 31 digits.
  - Bits and bit-field are arbitrary field lengths defined by the programmer as from 0 to 32 bits.
- *Special kinds of data* – These instructions are used extensively by the VMS operating system and include queue manipulation instructions that insert and remove queue entries, address manipulation instructions, and user-programmed general register load and save instructions.
- *Basic program flow* – These instructions include *branch, jump,* and *case* instructions, *subroutine call* instructions, and *procedure call* instructions.

- *Special operating system functions* – These instructions provide rapid and efficient rescheduling of the operating system functions. They include process control instructions such as context switching, which allows process context variables to be loaded and saved using one instruction for each operation. The find-first instruction allows the operating system to locate the highest-priority executable process.

- *High-level language constructs* – During the design of the VAX architecture, special attention was given to implementing frequently used, high-level language constructs as single VAX instructions. These instructions contribute to decreased program size and increased execution speed. Some of the single VAX instructions include:
  - The FORTRAN-computed GOTO statement that translates into the case instruction.
  - The loop construct such as *add, compare,* and *branch* that translate into the ACB instruction.
  - An extensive call facility that aligns the stack on a longword boundary, saves user-specified registers, and cleans up the stack on return. The call facility is used for compatibility among all native-mode languages and operating system services.

- *Addressing modes* – The VAX processors offer several addressing modes, many of which use the general registers to identify the operand location as PDP-11 addressing modes do. The modes are
  - Register
  - Register deferred
  - Autoincrement
  - Autoincrement deferred
  - Autodecrement
  - Byte, word, and longword displacement
  - Byte, word, and longword displacement deferred
  - Indexed
  - Literal

- GENERAL REGISTERS AND STACKS

The VAX processors provide sixteen 32-bit general registers that can be used for temporary storage, or as accumulators, index registers, and base registers. Registers R0 through R11 can be used as general purpose registers and the remaining four have special functions depending on the instruction being executed. The registers are R12 (argument pointer), R13 (frame pointer), R14 (stack pointer) and R15 (program counter). Stacks are associated with the processor's execution state. The processor may be in one of four process context modes–kernel, executive, supervisor, or user mode–or in the systemwide interrupt service context. A stack pointer is associated with each of these states. Whenever the processor changes from one state to another, register 14 is updated accordingly.

Software Overview

The VMS operating system and the ULTRIX-32 operating systems provide the basic system management and control. The operating systems organize and allocate the system resources and files to insure the protection of data and the efficient system operation.

- VMS OPERATING SYSTEM

The VMS system is a general purpose multiuser, multifunction operating system that is compatible with all VAX processors and provides reliability and high-performance for the concurrent execution of multiuser timesharing, batch, and time-critical applications. VAX/VMS features include the following:

- Virtual memory management for executing large programs.
- Event-driven priority scheduling.
- Shared memory, file, and interprocess communication data protection based on ownership and application groups.
- Programmed system services for process and subprocess control and interprocess communication.

The VAX memory management features allow swapping, paging, protection, and sharing of both code and data. Memory is allocated dynamically and applications can control the amount of physical memory allocated to executing processes, to the protection of pages, and to swapping. These controls can be added after the application is implemented.

CPU time and memory residency are scheduled on a preemptive priority basis. Therefore, time-critical processes do not compete with lower-priority processes for scheduling services. Scheduling rotates among processes of the same priority.

The VMS includes system services to control processes and process execution, time-critical response, and scheduling; and to obtain information. Process control services allow the creation of subprocesses as well as independent detached processes. Processes can communicate and synchronize using mailboxes, shared areas of memory, or shared files. A group of processes can also communicate and synchronize, using multiple common-event flag clusters.

Memory access protection is provided between and within processes. Each process has its own independent virtual address space, which can be mapped to private pages or shared pages. VMS uses the four processor-access modes to read-protect and write-protect individual pages within a process. Protection of files, shared pages of memory, and interprocess communication facilities such as mailboxes and event flags, are based on user identification codes that are assigned individually to accessors of data.

A complete program development environment is offered and supported by the VAX/VMS system. It includes the native assembly language, VAX MACRO, and the following high-level programming languages: VAX APL, VAX BASIC, VAX BLISS-16, VAX BLISS-32, VAX C, VAX COBOL, VAX CORAL 66, CORAL/VAX to RSX Cross Compiler, VAX DIBOL, VAX FORTRAN, VAX PASCAL, VAX PL/I, FORTRAN IV/VAX to RSX Cross Compiler, VAX LISP, and VAX RPG II. It provides the tools necessary to write, assemble, compile, and link programs, as well as to build libraries of source, object, and image modules.

Information Management–The VMS operating system supports a set of software tools that provides a full range of information management capabilities. These information management products can be used to organize, maintain, retrieve, and manipulate data quickly and easily. They include the following:

- *VAX ACMS* product set provides tools to develop and control complex online transaction processing applications.
- *VAX Common Data Dictionary* provides a single, logical data dictionary as a common source and contain descriptions for language and information management tools, including VAX DATATRIEVE and VAX ACMS.
- *VAX DATATRIEVE* is a multifaceted data management facility that can store, update, and retrieve information and generate reports.
- *VAX DBMS* is a CODASYL-compliant database management system for creating, maintaining, and updating databases.
- *VAX DSM* is a multiuser data management system and a high-level interpretive language for Digital Standard MUMPS (DSM).

- *VAX FMS* is a forms management system with interactive and language-callable video forms.
- *VAX Rdb/VMS* is a full-function, relational database management system for local and remote database applications operating under VMS software.
- *VAX Rdb/ELN* is a highly functional, relational database management system designed for the VAXELN operating environment.
- *VAX-11 RMS* is a record management facility for sequential, relative, and multikey ISAM (indexed sequential access method) file organizations.
- *VAX TDMS* is a terminal data management system and a programming tool for form-intensive applications operating with the VAX/VMS operating system.

The VMS on-disk structure provides a multiple-level hierarchy of named directories and subdirectories. Files can extend across volumes and can be as large as the volume set on which they reside. Volumes are mounted to identify them to the system. VAX/VMS also supports multivolume ANSI-format magnetic tape files with transparent volume switching.

Communications–The variety of communications interfaces supported by the VMS operating system allows VAX systems to be connected to other VAX systems, other Digital systems, and other manufacturers' computer systems.

Synchronous, point-to-point, and multipoint connections are supported for interprocessor communication. For terminal-to-host communications, asynchronous connections are supported.

Using DECnet communications software, various types of computer system networks can be constructed to facilitate remote communications, resource sharing, and distributed computation. *DECnet-VAX Phase IV* software offers adaptive routing, task-to-task communications, network file transfer, network command terminals, and network resource sharing and network management capabilities. *DECnet-VAX* also supports the connection of Digital's systems and terminals to local area networks (LANs) including Ethernet and to other manufacturers' systems through Internet applications. With *DECnet/SNA Gateway* software and hardware, communications can be established to IBM SNA networks. *DECnet Router/X.25 Gateway* connects DECnet nodes on an Ethernet to DECnet nodes on an X.25 packet-switched data network using the data-link mapping capability of the Gateway software. The *VAX X.25/X.29 Extension Package* allows remote terminals to access the host VAX processor on the Ethernet by "dialing in" through a network packet assembler/disassembler. It also allows access to the protocol level of X.25 traffic. The DECnet communication software supported by VMS includes:

- *VAX-11 2780/3780* is a protocol emulator for transferring data between a VAX system and an IBM or other manufacturer's system by means of remote job-entry protocols.
- *VAX-11 3271* is a protocol emulator for an interactive program-to-program link to an IBM host running CICS or IMS.
- *MUX200/VAX* is an emulator for communication with a CDC6000, CYBER series, or other host computer system capable of using 200 UT mode 4A communications protocol.
- *VAX PSI* is a packet system interface that allows a VAX/VMS system to connect to packet-switching data networks conforming to CCITT recommendation X.25.
- *IEC11-V* is a device driver that allows programs written in MACRO-32, FORTRAN, or BASIC to communicate with IEEE Standard 488 devices connected to IEC11-A and IEC11-B.
- LAT-11 is a special-purpose software system that enables a PDP-11 processor to be used as a local area terminal server on Ethernet.

- ULTRIX-32 OPERATING SYSTEM

The ULTRIX-32 operating system is a Digital-supported native UNIX operating system and provides complete UNIX functionality, enhanced especially for the VAX systems. The ULTRIX-32 software is an interactive, timesharing operating system derived from the fourth Berkeley software distribution (4BSD) technology developed at the University of California at Berkeley. ULTRIX-32 takes advantage of the VAX virtual memory architecture with demand paging and provides enhanced performance for applications requiring large amounts of memory.

In addition to features and performance comparable to 4BSD, Version 4.2, ULTRIX-32 provides the following benefits:

- Serviceability enhancements and reliability features.
- Repackaged technical documentation.
- Ability to install and tailor the system kernel for specific configurations without the need for sources.
- Support for VAX-11/780, VAX-11/750, and VAX-11/730 systems.

Some of the more familiar UNIX system tools for software development and text processing provided by ULTRIX-32 include:

- UNIX Version 7 Bourne and C Shells.
- Line and screen editors.
- File transfer capability.

- Ethernet support.
- Remote login.
- Remote job execution.
- Electronic mail.
- Text processing utilities.
- Programming languages, including C, FORTRAN 77, PASCAL, FransLISP, and UNIX Assembler.

VAX System Overview

VAX computer systems consist of the central processing unit, a console subsystem, the main memory subsystem, and the input/output (I/O) subsystems. The VAX processors are 32-bit, high-speed, microprogrammed computers that execute the normal instructions in native mode and nonprivileged PDP-11 instructions in compatibility mode.

Each VAX processor includes memory management hardware, sixteen 32-bit general registers, 32 interrupt priority levels, and a console subsystem linked directly to the processor. A cache memory is included with all the processors except for the MicroVAX I, VAX-11/725, and VAX-11/730 systems. A high-performance floating-point accelerator (FPA) is available or supplied with the VAX processors. The FPA operates in parallel with the basic processor to execute the standard floating-point instruction set. The FPA decreases instruction execution time of floating-point arithmetic and of some integer arithmetic operations.

- CACHE MEMORY

The cache memory provides the central processor with high-speed data access by storing frequently referenced addresses, data, and instruction items in the secondary memory. The memory cache significantly reduces the processor's effective memory access time. The cache memory in some of the VAX systems includes an instruction buffer, which enables the CPU to fetch and decode the next instruction while the current instruction completes execution.

An address translation buffer also is used to eliminate extra memory accesses during virtual-to-physical address translations. The address translation buffer contains frequently used virtual-to-physical address translations.

- CLOCKS

  The VAX systems include a programmable realtime clock used by system diagnostics and by the VMS operating system for accounting and scheduling, and a time-of-year clock, which ensures the correct time of day and date are used.

- MEMORY MANAGEMENT

  The VAX memory management hardware enables the VMS operating system to provide a flexible and efficient virtual memory programming environment. Hardware memory management and the operating system provide both paging with user control and swapping. The memory management hardware facilitates program and data sharing, and allows larger program size and increased performance.

- CONSOLE SUBSYSTEM

  The console subsystem allows the user to access the system and programs through the switches on the control panel, a console keyboard and printing terminal, the console command language, and an internal mass storage device. Simple console commands, entered through the console terminal, provide access to registers and operational control, which includes bootstrap program loading, initialization, and self-testing. The console subsystem and the console command language also facilitate the loading of diagnostics programs and software updates from the mass storage device. The console terminal is available to authorized users for normal system operations, thereby eliminating the need for a separate user terminal.

  The console subsystem also enables the remote diagnostic facility to communicate with the VAX processor. The remote diagnostic option allows the CPU to be connected through a modem to the Digital Diagnostic Center for fault detection or preventive maintenance. This option can significantly lower maintenance costs and contribute to the system's availability and maintainability.

- MEMORY SUBSYSTEM

  The main memory subsystem consists of a memory controller and memory array modules that use both 64-Kbyte and 256-Kbyte metal oxide semiconductor (MOS) RAM integrated circuits (ICs) for data storage. Each array module contains either 1 Mbyte or 4 Mbytes of memory. The error correcting code (ECC) allows the correction of all single-bit errors and the detection of all double-bit errors, ensuring data integrity. In the VAX-11/780 and VAX-11/785 systems, two memory controllers with equal amounts of memory can be interleaved to improve I/O throughput. In the VAX-11/782 system, the local memory is used only for stand-alone diagnostics.

The MA780 multiport memory option is available for the VAX-11/780 and VAX-11/785 systems and can contain a maximum of 2 Mbytes of ECC MOS memory. Two MA780 multiport memory options can be included with each processor, for a total of 4 Mbytes of memory. The multiport memory information is accessed by each system much as main memory is accessed and can be shared by up to four VAX-11/780 processors to providing common data access, high throughput, and increased availability in a multicomputer system.

The MA780 multiport memory with 4 Mbytes is part of a standard VAX-11/782 system configuration and is shared by the two VAX-11/780 processors. An additional MA780 multiport memory can be added to expand the main memory capacity to 8 Mbytes. The local memory provides only resident diagnostic programs.

- INPUT/OUTPUT SUBSYSTEMS

The input/output subsystems are the main communications paths between the VAX processors and terminals, mass storage devices, communication interfaces, VAXcluster devices, and customer-developed equipment. The UNIBUS is an asynchronous, bidirectional bus that provides the signal and control lines for connecting medium- and low-speed peripheral devices to the CPU. The MASSBUS consists of high-speed lines that permit the CPU to communicate with dedicated mass-storage devices. The computer interconnect (CI) is a UNIBUS adapter and controller that provides the local network connection between VAX processors and between VAX processors and intelligent shared-data storage devices through the VAXcluster networks. Redundant transmit and receive lines provide extremely fast and reliable information transfers between the CPU and storage systems connected to the VAXcluster network. The Digital Data Interconnect (DDI) is a high-speed, 32-bit, parallel data path and control path that allows direct access to the CPU memory from Digital or customer-developed devices.

▪ VAX System Dependability

Extensive reliability, availability, and maintenance features are an integral part of VAX systems. They ensure dependable operation of the systems and facilitate repairs when a failure occurs. With better system reliability, optimal system availability, and easier maintenance, uptime is increased and the overall cost of ownership is reduced.

During normal system operation, the system is continually monitored by both software and hardware to detect errors or malfunctions in the system operation. Depending on the type of error, the error can be either corrected by the hardware or software or reported to the operator. The VMS operating system records and reports errors to the operator through the console terminal.

Consistency and Error Checking

The continual checking of the consistency of instruction and information prevents errors from being repeated through a system database. These checks detect abnormal instruction uses, transient and permanent hardware errors, and illegal arithmetic conditions. Some of the specific checks include the following:

- *Arithmetic Traps*—These traps occur when overflow, underflow, and divide-by-zero arithmetic conditions are detected. The hardware detection of these error conditions is used by high-performance software. Overflow and underflow traps may be enabled or disabled by setting flags in the processor status longword, allowing the arithmetic exception conditions to be ignored if appropriate.
- *Limit Checking Traps*—The decimal string instructions have length limit checks (0 to 31 decimal digits) performed on the output strings to ensure that instructions do not overwrite adjacent data.
- *Reserved Operand Traps*—These are fields and opcodes that are reserved to customers and to Digital Equipment Corporation to ensure that the customer extensions to the VAX architecture that include user-defined instructions or data structures do not conflict with future Digital expansions.
- *Special Instruction Checks*—The *call* and *return* instructions have hardware-implemented register save/restore and consistency checking. These instructions provide a standard, identical interface for user routine calls and system calls. The cyclic redundancy check (CRC) instruction provides block-checking error code calculations that are important in communications applications.

The VMS operating system uses internal consistency checks to determine when data is not valid. These checks determine the validity of the system control information. System malfunctions cause a trap to an exception handling routine and the appropriate information is recorded in the error log file. The consistency checks include:

- *Exception Handling*—When any of the consistency checks detect a failure, the VMS software uses a uniform condition-handling facility to manage the hardware or software exception. Because the exception-handling facility is uniformly consistent within the VAX system design, operation is more predictable and reliable.

- *Error Correcting Code*—The memory error correcting code (ECC) automatically corrects single-bit memory errors and detects double-bit memory errors. The code will also detect greater than double-bit errors if an even number of errors are detected. The disk error correcting code detects most errors and corrects errors in a single error burst of up to 11 bits. The ECC provides protection from nonrepeating errors by automatically correcting data. Detections and corrections are noted in the error log as a preventive maintenance aid.

- *Mass Storage Verification*—The I/O verification for mass storage peripherals is supported by the VMS device drivers. The hardware compares each block for equivalence immediately after the block is read from or written to the device. The checking may be performed on all read or write operations to a file or volume, or for only a single read or write operation. This capability increases reliability of the read or written data.

- *System Verification*—The VMS operating system contains the user environmental test package (UETP), an automated collection of verification software. The UETP controls the comprehensive and systematic testing of peripheral devices and software components by performing most of the VMS utilities and language translators; by calling most of the VMS system services and I/O services, each of which has a wide range of parameters; and by comparing the results to known answers. During the execution, the system reports errors to the error log and to the console terminal. The UETP is a means of validating the installation and functioning of a VAX system. A user can execute the UETP to check system functions, to perform preliminary diagnosis, or to demonstrate system capability.

- *System Exerciser*—The system exerciser diagnostic program tests subsystems in a user environment operating under VMS software. It verifies hardware integrity or indicates subsystems that may be failing or undergoing deteriorating performance. This testing is performed automatically using online diagnostic programs. The system exerciser is part of an extensive hardware reliability verification procedure.

- *Automatic Online Error Logging*–Error logging, a software tool used to monitor error occurrences, is an integral part of the VMS operating system. The operating system accepts signals from the hardware and records CPU, memory, I/O, and software errors in an online log file. The error logger records information related to the state of the system at the time of an error. If no errors occur during a period of time, the time of day is recorded in the log file to indicate that the error logging process is operating. For ECC-corrected memory errors, if the error rate exceeds a threshold value, the ECC log entries are suspended for a period of time. A utility program is available to convert the log file into a format and summary that can be printed for later evaluation. Error logging is useful for the efficient maintenance of the hardware by providing a report that can help diagnose impending or persistent hardware problems. It detects trends in fault occurrences and helps to identify the specific subsystems, which can then be examined using diagnostics.

- *Machine Checks*–Machine checks are hardware errors detected by the processor and reported to the VMS operating system. The VMS software categorizes the error and takes appropriate action. The instruction in error is retried, and if the error is transient, the operation continues. If the instruction retry fails, the software attempts to limit the effects of the error to a single process. If the VMS executive is executing when the machine check occurs, the system is automatically rebooted. Machine checks are caused by cache, translation buffer, or control store parity errors or by failure of the I/O or memory subsystems to respond to CPU requests.

- *System Identification Register*–The system identification (SID) is a hardware register that maintains information pertinent to the system processor type and revision number. This information may be examined during the software error logging process to determine the engineering status of the processor.

- *Application Error Detection*–The VMS software, together with the optional software products, provides extensive checking for errors in application software. The parameters passed to the system services are checked before the service is attempted, thereby preventing the operating system and the application from being vulnerable to incorrectly coded programs. Tools such as the macro assembler and the linker provide a high level of error detection to minimize the creation of erroneous application software. This includes the detection of program syntax errors, corrupted files such as system libraries and user files, and inconsistency between separately compiled programs.

- *Deadlock Detection*–The VMS software provides lock manager services for the synchronization of multiple processes and for the queuing of processes depending on the availability of resources required. A critical problem can occur in a deadlock when each process is waiting for the completion of another process. The lock manager can detect this condition and prevent the application from being unusable.

Error Analysis and Recovery Features

The VAX system hardware and software determine the source of an error and the extent and impact on the user's operation. The system can often correct the error or mask its effects. Some of the analysis and recovery features are as follows:

- *System Dump Analyzer*–The system dump analyzer (SDA) is a VAX/VMS utility that helps to determine the cause of an operating system failure. When an internal error interferes with normal operations, the operating system writes information concerning its status at the time of failure to a predefined system dump file. The SDA examines and formats the content of this file. With the help of the SDA commands, a user can display parts of the formatted system dump file on a video display terminal, or can create a hardcopy listing. In addition to analyzing the system dump file, the SDA can perform operations on a system without interrupting system operation.

- *Error Log Reporting Program*–The system error analyzer is a VAX/VMS utility program that is available to convert the error log file into a format and summary that can be printed for analysis.

- *Instruction Retry*–When a hardware error, such as a machine check, interrupts the execution of an instruction, the system may reexecute the instruction. If the error is transient, then normal operation will continue. If the instruction cannot be retried or if the retry fails, the VMS software attempts to limit the failure to the user process currently being executed. If the operating system is executing, the system is automatically rebooted.

- *Nonfatal Bugchecks*–When an error condition is detected by hardware or software and VMS software determines that only a single process is affected, that process is removed from the system without affecting normal operation of the executive or other processes. The error is logged in the system error log.

- *Automatic Stack Expansion*–The VMS operating system automatically extends user stack space as needed.

- *Unattended Automatic System Restart*—The automatic system restart facility returns the system to an operating condition after a power interruption or a fatal software error. If the battery backup unit has preserved the contents of memory during the power interruption, the time-of-year clock allows the VMS software to recover the correct date and time of the power recovery. A special memory configuration register indicates to the recovery software whether data in memory was lost. Following a power failure, all I/O operations in progress before the failure occurred are restarted. The powerfail asynchronous trap facility may be used to initiate image-specific powerfail recovery processing. If the battery backup option is not installed or if the time of the power failure is longer than the battery backup can sustain the memory data, the system is rebooted when the power is restored. Following a system failure, unprocessed error log entries are written to the error log file and the contents of physical memory is written to the disk for later analysis. A switch on the control panel of the VAX processor enables the system operator to select automatic or manual restart of the system after a power failure.

- *Automatic Reconfiguration*—The VMS operating system allows the system to continue operating after some of the hardware components have failed. The system can be manually or automatically reconfigured at the startup time so that the system continues operation until the maintenance is performed. Some of the reconfiguring methods are as follows:
  - —If the program-loading device has failed, the program can be bootstrapped from a system disk on which the file can be loaded.
  - —If memory pages are defective, the memory is configured by the VMS system so that the defective pages are not referenced and are transferred to a bad-page list. This occurs during bootstrap loading and continues dynamically while the system is operating. A set of physical addresses is not required to bootstrap the system.
  - —If excessive cache errors are detected in the medium- and large-VAX processors, the cache memory can be dynamically disabled by the VMS software.
  - —The VMS operating system automatically determines the presence of peripherals on the processor at bootstrap time and flags nonresponding devices and memory as being unavailable.
  - —Software spooling allows the data output to be routed to an alternate device using system operating commands when the designated device is not available.

- *Dynamic Bad-block Handling*–Bad blocks of data may be generated when there is a failure of the disk drive that performed the data transfer. When the hardware detects a bad block during an I/O operation, the VMS operating system marks the header of the file in which the error occurred. When the file is deallocated, the system checks the file header for bad blocks and designates this information as permanently in use and not available for use by other files.

- *Mass-storage Error Recovery*–The operating system attempts recovery from nonfatal disk and tape errors. If a hardware error occurs during an I/O operation, the I/O operation is reinitiated by the software using the available hardware recovery mechanisms.

- *Redundant Recording of Critical Disk Information*–Critical disk information, such as the home block and the index file header, is duplicated to allow its recovery when it is accidentally destroyed.

- *Selective Hardware Disabling*–Several hardware elements, including the memory management logic, cache memory, translation buffer, and floating-point accelerator, can be disabled by the diagnostics program to help in isolating hardware failures. The translation buffer and cache memory are also dynamically disabled by the VMS program to allow the system to continue to operate at a reduced level of performance.

Data Integrity

The VMS operating system prevents interference between processes that contain critical system data. This is accomplished by the following hardware access protection and software enforced privileges:

- *Memory Management Hardware*–The memory management hardware assigns priorities to the kernel, executive, supervisor, and user modes of operation. The critical software components of the VMS software are run in the most privileged kernel and executive modes. Access to these modes is allowed only to authorized users; therefore, the integrity of the information is ensured.

- *Quotas and Privileges*–The VMS software uses a system of quotas to protect shared system resources, such as dynamic memory and page file space. Quotas are assigned on a per-process basis so a process cannot stop normal system operation by depleting shared resources. Hardware access protection and a set of software enforced privileges prevent user processes from affecting each other or the operating system. When a process violates the protection rules, the error is reported so that appropriate recovery actions can be taken without affecting the remaining system.

- *Access Control to Files and Volumes* – The VMS operating system provides protection on a per-file basis. Users can be assigned read or write access to other users' data and files by limited authorization on a volume-wide basis.

- *Disk Volume Protection* – The VMS file system allows the system manager to specify the amount disk space assigned to a user. This prevents a user's application from using all the available data storage. When a volume is mounted, the file structure is checked and any inconsistencies that exist are corrected automatically. This feature is useful when a system failure occurs while the files are in use. When a power failure to the system or disk drive occurs, the VMS file system checks to ensure that the volume in use at the time of the failure is the one to be reinstated. This prevents the accidental corruption of a different volume.

Maintenance Aids

The VAX system hardware packaging and diagnostic tools reduce the time necessary to diagnose and repair the system, thus increasing system availability. Some of these features are as follows:

- *Improved System Packaging* – The VAX systems components are designed for reliability and are easily accessible for repair. Most internal cables and assemblies are permanently mounted and accessible from the front or rear of the cabinet. This prevents damage that could be caused when assemblies mounted on slides are repaired.

- *Environmental Conditions* – The ac and dc power conditions, cabinet temperatures and internal cabinet air flow are monitored by sensors to detect abnormal environmental conditions. This prevents equipment damage and helps to determine the cause of failure.

- *Online Functional Diagnostics* – Many functional diagnostics can be performed online under control of the VMS operating system. This enables the isolation of malfunctions while the system is operating. The following functional diagnostics are implemented.

  – Character printers, line printers, card readers, and terminals.

  – Synchronous link and bus interaction.

  – Disk formatter and disk and tape reliability.

- *Fault-isolation Diagnostics* – Once the functional diagnostics have isolated faults in a particular subsystem or device, fault-isolation diagnostics can be performed to localize the failure to the smallest possible element. Fault-isolation diagnostics operate independently offline and are used to isolate problems to a field-replaceable unit. This minimizes the cost of replaceable parts and reduces the time of repair. Microdiagnostics can be performed by a Digital remote station when the remote diagnostic service has been contracted.

- *Online Software Update and Maintenance* – The system operator can perform software updates and maintenance activities while the system is actively performing operations. Software updates are distributed in machine-readable form on disk or tape media. During normal system activities, software modules on disks can be updated using patches and replacement modules, and software maintenance and disk backup procedures can be performed. This increases system availability and maintainability.

- *Automatic Updates and Patches* – The VMS software implements a system of automatic updates and patches. If a power interruption occurs when software updates are performed, the operation can be resumed when the system is restarted. Each patch automatically checks for previous patches and updates the current revision number. The patches are distributed and applied in machine-readable form, which eliminates errors that can occur during transcription.

- *High Resolution Interval Clock* – The VMS software and the interval clock are used by diagnostics programs to test time-dependent functions; therefore, machine-specific timing loops in programs are not required.

- *Maintenance Registers* – These registers contain specific maintenance information that can be examined when an error occurs to help determine the cause of the failure.

VAX-11/725 and VAX-11/730 System Features

The VAX-11/725 and VAX-11/730 systems contain several features that facilitate maintenance and ensure maximum availability of the system. The system includes diagnostic programs that can be performed by the customer to verify the operation of the system. Microverify routines are performed automatically during the powerup sequence, when the program is bootstrap loaded, or when the reset switch on the control panel switch is activated. These routines test the data paths, and the successful execution of the routines indicates that the system will bootstrap properly. Parity checking is performed on words read from the writable control store (WCS) before execution.

- CUSTOMER DIAGNOSTICS

Diagnostic programs can be initiated by the user to verifying that the hardware is operating properly and to isolate hardware failures when they occur. The error information provided by the diagnostics can be analyzed by Digital service personnel and spare parts and tools can be provided to repair the failure. Most failures can be corrected by one visit to the customer.

The user diagnostics provide error information in the following three modes of operation:

- The *autotest* mode is used to verify the operation of the CPU, the VMS system disk, and the diagnostic load disk. It can be used to determine the cause of a nonbootable system and is invoked by the *test* command in response to a console mode prompt. Messages are displayed during the test to indicate test progress. The offline menu mode is initiated after the successful completion of autotest.

- The *offline menu* system allows a user to verify the operation of any device on the system. The concise menu format provides a simple interface for users that are unfamiliar with VAX diagnostics. All standard VAX-11/725 and VAX-11/730 system options and devices are supported. When necessary, the user is prompted to supply necessary device preparation.

- The *online menu* system allows the user to verifying the operation of a device while operating under the VMS program. Device testing is performed without disturbing other users or processes on the system. Terminals, lineprinters, and other peripheral devices that do not affect the overall operation of the VMS program can be tested using the *online menu* mode. Devices that have both online and offline support should be checked with both menu modes for total device verification.

Remote Diagnostic Service–DECservice is Digital's field service operation for customers. Remote technical support is provided when necessary to the Digital service engineer at a customer site. DECservice customers also have the added benefit of a service feature called remote hardware monitoring (RHM). Under RHM, Digital Field Service will determine a schedule with the customer for performing periodic remote hardware checks of the system. This program allows Digital to identify potential problems and schedule preventive maintenance before a critical situation can occur.

- PROCESSOR HARDWARE

  The CPU contains the following hardware features to enhance the reliability and maintenance of the system:

  - *Module Replacement*–The system memory arrays, CPU modules, floating-point accelerator module and I/O interface modules can be easily removed and replaced.
  - *Modular Power Supply*–The power supply for the systems contain three modules: a power controller and two power regulators. These modules can be replaced in a short time. Status lights on the supply indicate the condition of the supply voltage.
  - *Dual TU58 Tape Drives*–If one of the TU58 tape drives is inoperative, the remaining drive can be used to load the program information.
  - *Air Flow Sensors*–Sensors are included to detect the movement of cooling air through the cabinet. Changes in the air flow are sensed and reported.

VAX-11/750 System Features

The VAX-11/750 system includes many hardware and software features that increase the system reliability and ensure efficient maintenance. Microverify routines are automatically performed during the powerup sequence, when the program is bootstrap loaded, or when the reset switch on the front panel is activated. These routines test the data paths, the 16 general-purpose registers, most internal CPU registers, the instruction prefetch buffer, parity logic of the cache memory, the translation buffer, and the cache memory. The successful execution of the microroutines indicates that the operating system can be properly loaded.

Parity checks are performed on MASSBUS data, control and address translation, UNIBUS address translation, memory cache data and address, address translation buffer transactions, microcode, and user control store information.

- REMOTE DIAGNOSTIC SERVICE

  Remote diagnostic service is available to customers in North America and parts of Europe. When the service is contracted, a remote Digital diagnostic center can test the operation of the system and an experienced staff can analyze the system problems. These services are available seven days a week, and 24 hours a day. The center has a host diagnostic computer that aids in localizing system failures. After a call is received from a customer, a technician at the center makes contact with the customer's computer and

logs onto the system. A series of tests and diagnostic procedures are performed from the service center to examine memory locations and the error log file. If servicing is necessary, the repair time is substantially reduced because the service engineer will have the required part when arriving at the site. Some of the programs can be executed while the system is processing customer information. Preventive maintenance is also offered through remote diagnosis centers and can be scheduled for convenient off-peak periods.

A diagnostic console is included with the remote diagnosis service and can access the processor's major buses and key control points through a special internal diagnostic bus. The console enables the operator to initiate the diagnostics programs using simple keyboard commands. The diagnostic console can also serve as an operator console and as a user terminal. The diagnostic console includes a cartridge tape drive and hardcopy terminal.

- PROCESSOR HARDWARE

The CPU contains the following hardware features to enhance the reliability and maintenance of the system:

- *Module Replacement*–The system memory arrays, CPU modules, floating-point accelerator module, and I/O interface modules can be easily removed and replaced.
- *Modular Power Supply*–The power supply for the systems contain three modules: a power controller and two power regulators. These modules can be replaced in a short time. Status lights on the supply indicate the condition of the supply voltage.
- *Improved Air Flow and Design*–Air is drawn into the top of the cabinet by the blower, passed through the modules and power supplies, and is exhausted at the bottom rear of the cabinet. The blowers operate well below their maximum ratings, which significantly extends their operating life. The cooling air flow is adequate when the cabinet doors are open, thereby allowing the online servicing of the units.

VAX-11/780, VAX-11/782, and VAX-11/785 System Features

Many features are included with the VAX-11/780, VAX-11/782 and VAX-11/785 processors to ensure reliable operation of the system and to make them easy to maintain. A complete set of diagnostic programs is available to test the main functions of the processor logic, memory arrays, I/O subsystems and devices.

When the UNIBUS adapter detects certain error conditions on the UNIBUS, the conditions are reported in the error log. If the conditions persist, the UNIBUS is reinitialized and the I/O operations currently in progress on the UNIBUS are restarted. The power to the UNIBUS adapter and to the UNIBUS peripheral cabinet can be removed and applied without affecting the normal system operation. This allows the replacement of faulty devices without interrupting the processor operation. The operations in progress on other UNIBUS devices are restarted automatically when the UNIBUS powered is restored.

The synchronous backplane interconnect (SBI) includes a 16-level silo register that monitors the central bus activity and records the status and parity check information of the 16 most recent cycles of bus activity. If an error or predetermined special condition occurs, the silo stops operating and the contents of the silo and can be examined to determine the cause of the failure.

The clock rate of the central bus can be varied by commands issued from the console terminal. This feature helps service engineers to diagnose intermittent hardware failures.

Parity checks are performed on MASSBUS data, control and address translation, UNIBUS address translation, memory cache data and address, address translation buffer transactions, microcode, and user-control-store information.

- REMOTE DIAGNOSTIC SERVICE

Remote diagnostic service is available and provides the same features as those described for the VAX-11/750 systems. The diagnostic console, included with the systems, can access the central processor's major buses and key control logic through a special internal diagnostic bus. Through the console, the operator can initiate diagnostic operations using simple keyboard commands. The diagnostic console also can be used as an operator console and as a user's terminal. It includes an LSI-11 microcomputer, floppy disk drive, and hardcopy terminal. A watchdog timer in the LSI-11 detects a stalled processor condition such as a failure to fetch an instruction.

- PROCESSOR HARDWARE

The CPU contains the following hardware features to enhance the reliability and maintenance of the system:

- *Module Replacement*–Memory arrays, CPU modules, floating-point accelerator modules and I/O interface modules can be easily removed and replaced.

- *Modular Power Supply*–The power supplies for the system are modular and can be easily removed and installed. Spaces are provided within the cabinet for six power supplies; four are part of the basic cabinet requirements and the remaining two are installed with the floating-point accelerator option and with the MASSBUS adapters option.
- *Improved Air Flow*–Three blowers, mounted at the top of the processor cabinet, provide the cooling air for the processor. The fans draw air into the top of the cabinet, pass it through the modules and power supplies, and exhaust it from the bottom rear of the cabinet. Because the air is supplied from the top of the cabinet, the air is not contaminated with dirt particles from the floor. The blowers operate below their maximum ratings, which significantly extends their operating life. The cooling air flow is adequate when all cabinet doors are open to enable the online servicing of internal units.

VAX 8600 System Features

In addition to the features included in all VAX processors, the VAX 8600 system incorporates many other features to ensure reliable operation, system availability, ease of maintenance, and maximum performance. The hardware and software of the VAX 8600 incorporate the latest design techniques to assure efficient operation and maximum system dependability. Approximately 15 percent of the logic circuits in the system are used to verify the operation of the system and the integrity of the data. More than 3,000 logic points on the CPU backplane are available for analysis through the serial diagnostic (SD) bus and the console terminal. Online diagnostic programs can be performed concurrently with the user mode of the VMS operating system without interrupting normal CPU operations. If errors occur during the system operation, they are recorded and logged. The CPU is restarted automatically and allowed to continue processing the instruction. Some of the salient reliability, availability, and maintenance features are:

- Fault-tolerent design.
- CPU signal levels that can be monitored through the console terminal.
- Error correction control (ECC) implemented on memory, cache, and CPU microprocessor control logic.
- Online diagnostics for the VMS operating system.
- System-error detection and logging.
- Autodiagnostic and remote diagnostic capabilities.
- Special maintenance capabilities provided by the console processor functions.

- FAULT-TOLERANT DESIGN

  The VAX 8600 processor incorporates a fault-tolerant design that allows the system to rapidly recover from system errors by monitoring status information related to the error, by recording the information in an error log, and by reinitiating the instruction in process when the error occurred. When an error is detected that results in a machine check, up to 88 bytes of error status information can be recorded. Since most errors that occur on processors using LSI logic components are a result of intermittent failures, the processor is allowed to continue operating after the error information is stored. If the error is repeated, the stored information can be evaluated later and maintenance can be performed at the convenience of the customer.

- DYNAMIC FAULT INSERTION

  Dynamic fault insertion allows the fault detection logic to be checked while the VMS software is operating. Intermittent faults can be simulated by selecting the desired timing of the CPU clock, and by inserting and removing a fault at the proper interval during one or more cycles. This is used to verify that the recovery logic included with the CPU is being executed correctly.

- SERIAL DIAGNOSTIC BUS

  The serial diagnostic (SD) bus is a network data path between the console subsystem and the CPU that connects to the major logic elements in the CPU. The SD bus is used to verify the system operation, diagnose and isolate CPU hardware faults, and provide a path from the console subsystem to control the operations of the CPU. The SD bus and the SD bus interface allow internal signal levels and signal levels transferred between modules through the system backplane to be accessed and displayed on the console terminal. This eliminates the need for the special test equipment normally required to monitor CPU logic levels during maintenance. The SD bus includes independent clocking and control, thereby enabling the console software to initialize the CPU hardware, step the CPU clock, monitor the state of various CPU logic signals, and serially transfer data to the console where it can be verified. By stepping the CPU clock, the console software is allowed to follow the progress of the CPU operation and verify the logic response through the SD bus. The SD bus consists of 24 serial data paths that connect to 24 separate visibility control channels included with the Fbox, Ebox, Ibox, and Mbox. A visibility channel in a CPU module consists of an 8-bit visibility register and a multiplexer that selects internal logic information to be loaded into the visibility register. The console program reads a visibility channel by stopping the CPU clock and by shifting an 8-bit address code to select a multiplexer channel. The register information is then transferred serially from the visibility register to the console subsystem and displayed on the console terminal.

- ENVIRONMENTAL MONITORING

The VAX 8600 processor includes a microprocessor-controlled environmental monitoring module (EMM) in the modular power supply area of the cabinet that samples and controls the environmental conditions in the CPU and optional system cabinets. The module monitors and controls the dc regulator voltages and checks the regulator error status. It also monitors the cabinet air flow, cabinet air temperatures, and ac ground currents for excessive current flow. The functions of the EMM are monitored and controlled by console commands from the console subsystem. By controlling the dc regulator voltages from the console terminal, marginal voltage conditions can be established to help detect intermittent logic errors.

- ERROR ANALYSIS AND REPORTING

The standard package of error analysis and reporting (SPEAR) program allows system maintenance to be deferred until a time is convenient for the customer. The program is based on the number of errors that occur and the customer is automatically notified when error rates exceed a specified value. SPEAR is a software maintenance tool that contains algorithms used to analyze the machine check data that is collected by the processor over a period time. Intermittent failures that allow the processor to recover and continue operation are stored in an error file and the file is examined by SPEAR to determine the cause of the fault. The program detects specific patterns related to one error or to a series of errors.

- PARITY CHECKS

The validity of the data and control information transferred between the processor elements is ensured by parity checks and a comparison of the stored information with information stored in another location. Most critical information has a parity indicator appended to it during the transfer. The checks performed on the VAX 8600 processor elements include the following:

- *Cache Memory Parity Checks* – When single-bit errors are detected during data transfers from the cache memory, the data is transferred with an indicator specifying that the data contains an error. A cache correction cycle is initiated to correct the single-bit error and to rewrite the corrected data into the cache memory. The logic element that requested the data issues a processor-interrupt request and a machine check stack frame is generated. The instruction is reexecuted and the corrected data is then accessed.

- *Control Store Parity Checks*–Single-bit errors associated with the control store logic of a controlling microprocessor in the CPU can be corrected. When a control store parity error is detected, the CPU is halted and the data containing the error is transferred to the console subsystem and corrected by the console ECC software. The micro word is then rewritten into the control store location and the instruction is automatically reissued without causing a fatal halt condition of the CPU. When an Mbox parity error is detected, the bit in error is identified and recorded; however, the CPU will not recover and resume operations.

- *General Purpose Register Checks*–Single-bit parity errors associated with the general purpose registers (GPRs) are correctable. Because of the multiple sets of GPRs in the VAX 8600, if an error is detected in the information stored in one GPR set, a copy of the information from a valid GPR is written into the GPR that contains the error.

- *Internal Bus Parity Checks*–Parity is generated on the information transferred through the internal processor buses. The parity is produced and checked on 8-bit bytes and on 32-bit words on some of the buses.

- *Ebox Arithmetic Unit Parity Checks*–The Ebox contains three sets of arithmetic logic units (ALUs) that receive the same inputs and perform the same functions. If the outputs of the three ALUs are not the same, a failure is recorded and the instruction is executed again.

- *Control RAM Parity Checks*–Parity checks are performed on the outputs of the control RAM to insure that the control functions are not in error.

- PROCESSOR HARDWARE

The design of the CPU hardware includes many reliability and maintenance features including the following:

- *Socket-mounted RAMS*–The RAM integrated circuits used on the VAX 8600 modules are mounted in sockets for easy removal and replacement. This feature eliminates the need to replace the entire module when a defective RAM is located, resulting in lower maintenance costs and faster repair.

- *Logic Partitioning*–Logic functions are grouped together on modules so that when a module is replaced due to a suspected failure, all the logic associated with that function is replaced.

- *Cabinet Cooling*–The CPU cabinet allows conditioned air, ducted through a raised floor, to enter through the bottom of the cabinet. The air passes through the modules and the modular power supplies and is expelled through a plenum at the top rear of the cabinet. The plenum chamber is muffled to keep the overall noise level within the acoustic limits of 60 decibels per ampere.

Chapter 2 · VAXcluster and Network Architecture

An important consideration in choosing a computer system is its ability to be expanded to support new users and applications. It may be necessary, initially or later on, to add main memory, increase the processing power, add mass storage devices, or extend communication with other systems and terminals. System expansion should be economical, without requiring costly changes in the processor architecture and software.

Digital's unified VAX computer architecture, network architecture, and mass-storage architecture provide the means for system growth to meet present or future expansion requirements. VAX computer architecture ensures that existing and new VAX processors are compatible and will operate effectively without extensive changes to software or hardware. VAXcluster and network architecture offers extensive capabilities for connecting computers and terminals – located in the same areas or dispersed to locations around the world – into efficient and flexible communication networks. Computing power, storage capacity, and system availability can be easily and economically increased. As new computers and devices are developed, they can be incorporated into the existing VAX systems without the need to replace software or equipment.

The VAXcluster architecture enables the high-speed interconnect function of the Digital Storage Architecture (DSA), implemented by the Computer Interconnect (CI) and the VMS software, to share computer resources among the cluster nodes, to increase system performance, and to ensure the availability and integrity of the system data. When system expansion is necessary, the CI provides the most efficient and cost effective means of adding CPU power and mass-storage devices. The expansion is accomplished easily and incrementally as the system requirements evolve.

Through the DSA, intelligent controllers, I/O servers disk and tape drive can be added to a system. One system is designed to be used with a single VAX processor and the remaining subsystem is for multiple-processor systems. Each subsystem contains a mass storage control and management functions are performed by the intelligent controllers. The UDA50 subsystem controller is used with the RA60, RA80, and RA81 disk drives, and the HSC50 intelligent mass storage server is used with the same disk drives and with the TA78 tape drives. Each controller is microprocessor-based and coordinates the activities of all storage drives that are connected to the

controller. The controllers are responsible for optimizing data throughput between its drives and the host processor. It optimizes each I/O request based on the positioning of the drive heads to effect the most efficient data transfers. The controller enhances data integrity by validating each transfer and performing error recovery when necessary without assistance from the host processor. A sophisticated error correction code is used to correct extensive data errors, thereby substantially reducing the loss of data when an error occurs.

VAXcluster Systems

VAXcluster systems provide a means for VAX customers to increase computing power, storage capacity, and system availability. A VAXcluster system can include one or more VAX-11/750, VAX-11/780, VAX-11/785, and VAX 8600 processors and intelligent mass storage servers connected to a common distribution unit (SC008 Star Coupler) by the Computer Interconnect (CI) bus as shown on figure 2-1. VAXcluster configurations are fully supported by the VMS operating system. Processors and intelligent mass storage subsystems are connected into a loosely coupled local network that enables the sharing of processing power and data resources. VAXclusters are implemented by the high-speed, dual transmission CI path for the transfer of information between the processors and the HSC50 mass storage servers.

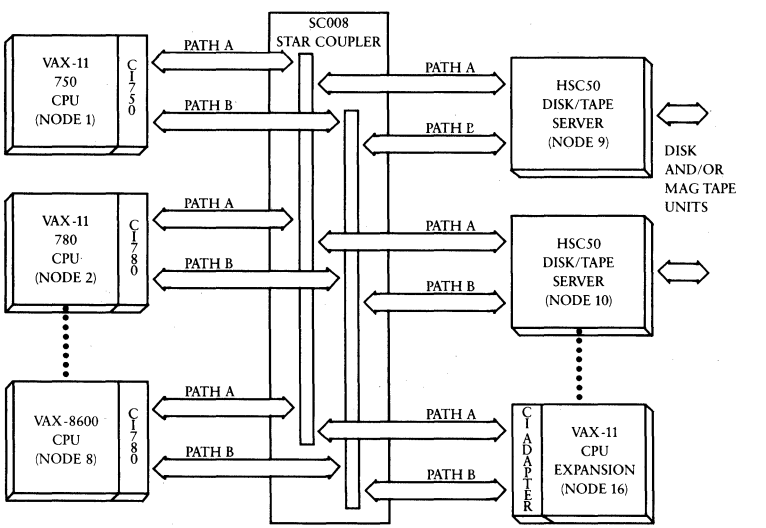


Figure 2-1 ▪ VAXcluster System Configuration

The features and benefits of the VAXcluster systems are as follows:

- Processing power and data storage capacity can be added incrementally to the system.
- The choice of VAX processors and mass storage facilities is based on individual system requirements.
- Each VAX processor operates independently to increase the total system reliability and availability.
- High-speed message-oriented transmission is provided.
- Intelligent mass storage servers and VMS facilities ensure data integrity and file sharing.
- Dual-path computer interconnect provides intelligent, high-performance redundant lines.
- Common access from any VAX system is provided to information stored in data files.

Computer Interconnect

The Computer Interconnect (CI) is a dual-path, 70-Mbits-per-second-bandwidth, logically multidropped link used to transfer serial data between local VAXcluster nodes. The CI can connect up to 16 nodes and a node can be a VAX processor or an HSC50 mass storage server. Interprocessor communication is controlled by new system-level protocols. The physical connection to the VAX systems is through the CI interconnect adapters and coaxial cables. The adapters are microprocessor-controlled, intelligent interfaces that perform the functions of fully buffered communication ports. Messages are transferred in blocks of data between the CPU memory and other nodes within the VAXcluster by the VMS and DECnet-VAX software. By providing the necessary data buffering, address translation, and encoding and decoding, the CI adapters significantly reduce the amount of overhead processing required to complete high-level intercomputer communication. The CI consists of an adapter and two transmit and two receive coaxial cables that attach to the SC008 Star Coupler unit. Information can be transferred simultaneously on either of the two transmit or receive lines. Two CI adapters can be installed on the VAX-11/750, VAX-11/780, VAX-11/785 and VAX 8600 processors to increase performance and ensure reliable data transmission in the event of a failure in a CI adapter.

The CI750 adapter is the interface between the CPU/memory interconnect (CMI) of the VAX-11/750 processor and the dual-path CI bus. The CI750-BA or CI750-BB is mounted in a cabinet 101.6 centimeters (40.0 inches) high and contains three extended-length hex-height modules, backplane mounting unit, and power supply for an input of 120 volts (ac) or 240 volts (ac). The CI750 connects to the VAX-11/750 through a hex-height interface module that mounts in the processor backplane. The CI750-SA or CI750-SB consists of two CI750-BA or CI750-BB options, cables, and SC008 Star Coupler unit for operation with two VAX-11/750 processors. Figure 2-2 shows the CI750 cabinet.

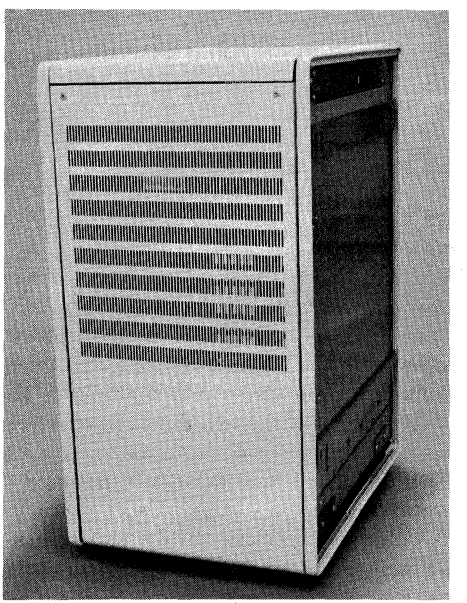


Figure 2-2 ▪ CI750 Computer Interconnect Adapter Unit

The CI780 adapter is the interface between the synchronous backplane interconnect (SBI) bus of the VAX-11/780 series of processors or the VAX 8600 processor and the dual-path CI bus. The CI750-AA or CI780-AB options can be installed in the CPU cabinet of the processors or in a CPU expansion cabinet. It consists of four extended-length, hex-height modules, backplane mounting unit, and power supply for an input of 120 volts (ac) or 240 volts (ac). The CI780-SA or CI780-SB consists of two CI780-AA, -AB options, coaxial cables, and the SC008 Star Coupler unit for operation with one processor or with two separate processors. Figure 2-3 shows the CI780 option hardware.

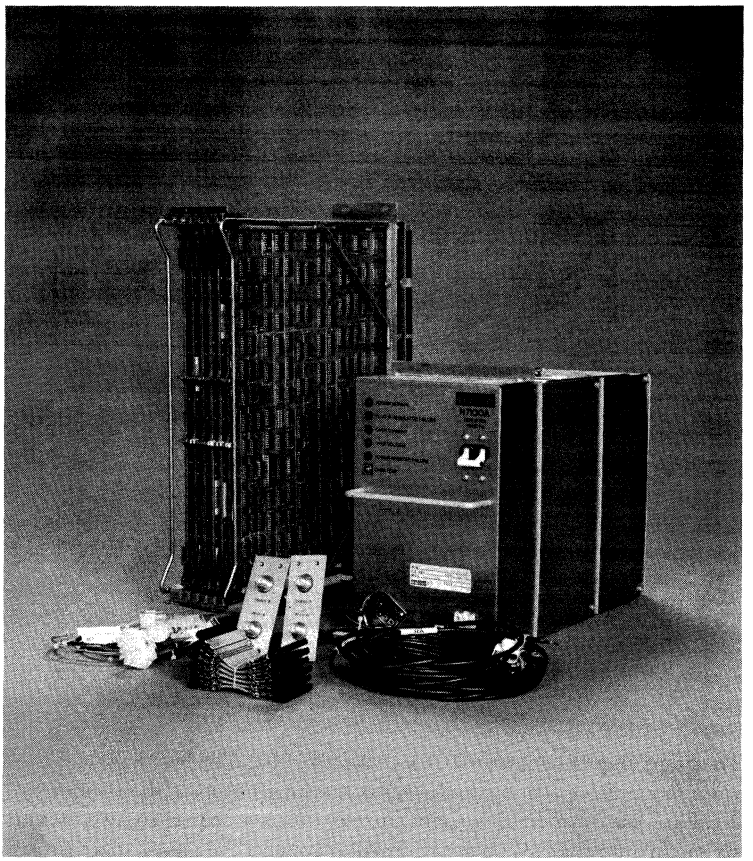


Figure 2-3 • CI780 Computer Interconnect Adapter Hardware

SC008 Star Coupler Unit

The SC008 Star Coupler unit, shown in figure 2-4, is the common connection point for all the coaxial cables from the VAXcluster nodes. It terminates and electrically isolates the serial-data lines and distributes the data to and from the individual nodes. The Star Coupler is an RF, transformer-coupled, dual-path, passive unit that can connect with up to 16 CI nodes. The maximum length of the coaxial cables that connect a node to the SC008 unit is 45 meters (147.6 feet).

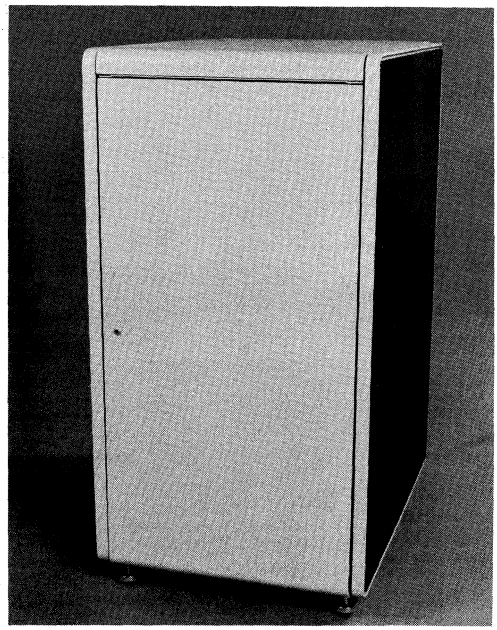


Figure 2-4 ▪ SC008 Star Coupler Unit

The Star Coupler cabinet is 101.6 centimeters (40.0 inches) high and provides mounting space for up to four connecting panels as shown in figure 2-5. Each panel has eight transmit and eight receive coaxial connectors and six modularity coaxial connectors used to connect the panels together. Up to eight nodes can be connected to the two-panel configuration and up to 16 nodes can be connected to the four-panel configuration. Nodes may be added or removed without interrupting the operation of the other nodes. The coaxial cables are routed through the rear of the cabinet.

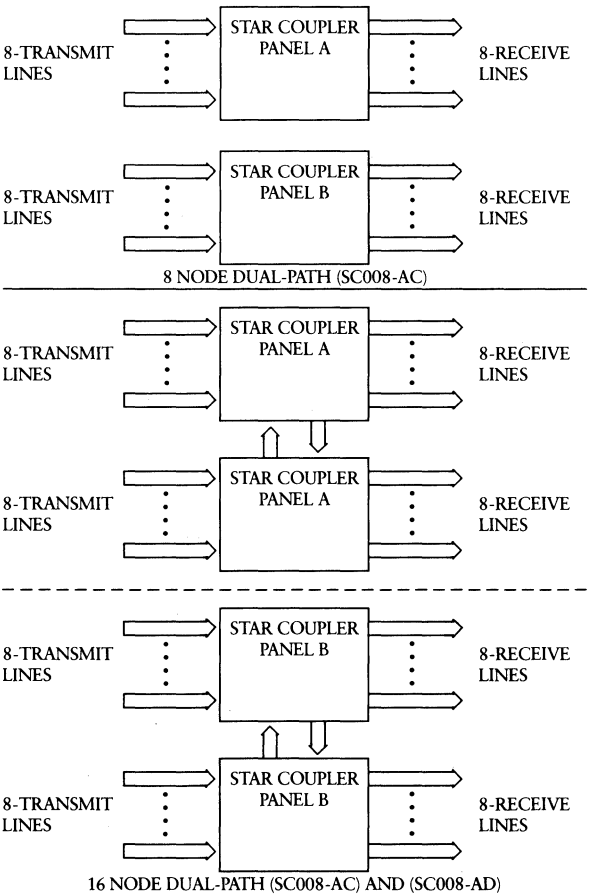


Figure 2-5 • SC008 Star Coupler Terminating Panel Configuration

HSC50 Intelligent Mass Storage Server

The HSC50 (Hierarchical Storage Controller), shown in figure 2-6, is an intelligent mass storage I/O server that connects the VAX processors in a VAXcluster to a group of mass storage disks or tape drives. The HSC50 server and drives form a node on the VAXcluster system. Operating with VAX/VMS software, it supports up to 24 Digital Storage Architecture (DSA) devices including the RA60 205-Mbyte removable-disk drive, the RA80 121-Mbyte fixed-disk drive, the RA81 456-Mbyte fixed-disk drive, and the TA78 magnetic tape subsystem. The disk drives contain two communication ports and each port can connect to a HSC50 server. Communication between the HSC50 server and the mass storage devices is through the standard disk interfaces (SDI) and the standard tape interfaces (STI). High-speed, I/O throughput is performed under control of a PDP-11 processor operating with a multiple high-speed bit-slice microprocessors. By adding multiple HSC50 servers in a VAXcluster system, higher data throughput and system performance can be achieved.

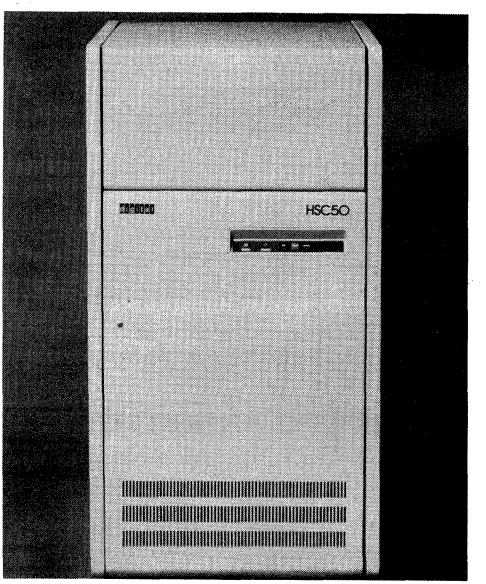


Figure 2-6 ▪ HSC50 Intelligent Mass-storage Server Unit

The HSC50 server performs many disk management functions, including volume shadowing, volume image copying, and image backups from the host, thereby reducing the load of the host VAX processor. The database is shared between the VAX processors connected to the VAXcluster architecture. The HSC50 server manages the physical activities of the drives, optimizes the system throughput, detects and corrects errors and performs local functions such as diagnostic test execution without the intervention of the VAX processor. Data is transferred between the HSC50 server and the host VAX processor using a series of communication protocols, all conforming to the MSCP (Mass Storage Control Protocol) of DSA. Its internal bus structure allows read and write operations to be performed simultaneously to the disk drives. The HSC50 server includes a comprehensive set of diagnostic programs and redundant logic to ensure reliable performance and to facilitate maintenance. The capabilities provided by the HSC50 server and devices are as follows:

- Command queue stores up to 120 host processor I/O requests
- Provides data buffering of up to 128 Kbytes for 256 disk sectors to optimize data flow
- Self-contained diagnostic test programs
- Offline utilities provide volume backup and restore as well as disk formatting and format verification
- Dual-ported disk drives can connect to two HSC50 servers for higher availability
- Multiple VAX processors can read from or write to the shared files.

The HSC50 server includes controller logic, two TU58 cartridge tape drives, and a power supply mounted in a cabinet 101.6 centimeters (40.0 inches) high. One TU58 tape drive is accessible from the front of the cabinet and one is in the cabinet. A distribution panel mounted at the rear of the unit is used to connect the cables from the disk and tape drives.

The configuration of the HSC50 server and devices is shown in figure 2-7. It connects to the SC008 unit through the CI coaxial cables and to the tape formatters and disk drives through six data channels.

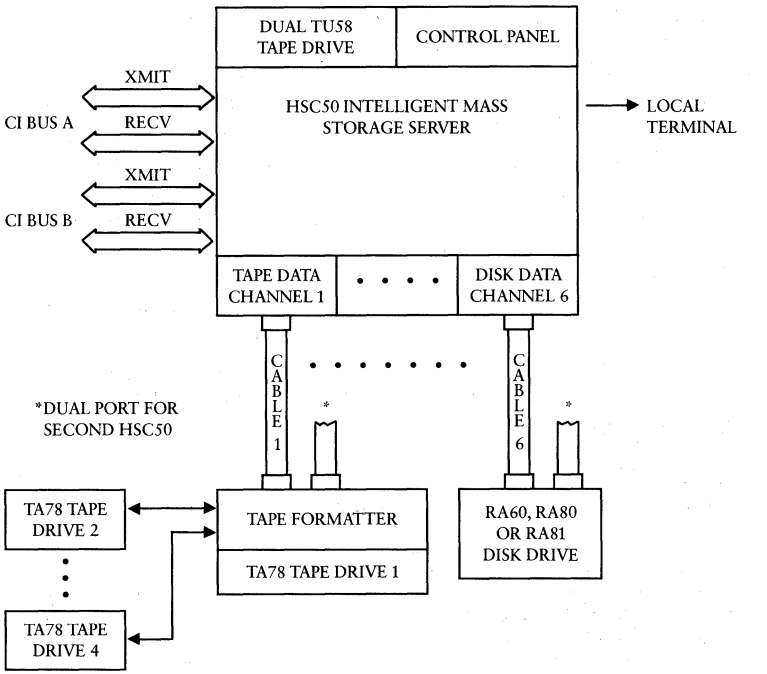


Figure 2-7 ▪ HSC50 Mass Storage Server Configuration

Communication Networks

Digital offers extensive capabilities to link computers and terminals into network configurations. Networks can increase the efficiency and cost-effectiveness of data processing by allowing computer systems and terminals to share resources and to exchange information files and programs. Applications can be expanded by adding computers and terminals to the networks as required. Through the use of networks, small computers can have access to the capabilities of larger, more powerful systems and the large systems can have access to the information processed by the dedicated applications of the smaller systems.

Communication networks can be installed within a building, or within a site between buildings. By the use of a common carrier of dedicated telephone lines, the communication facilities can be extended to remote locations throughout the world.

The Digital Network Architecture (DNA) is the framework for all Digital network products. It is a comprehensive set of hardware and software used to control the communication functions of the networks. The Digital System Interconnect (DSI) strategy provides the process that allows VAX processors operating with VMS software to communicate with mass storage devices, terminals, and communication links. The DNA is closely based on the International Standards Organization (ISO) architecture for open system communication. These industry standards allow systems developed by different manufacturers to be compatible. The DNA offers the flexibility of adding new hardware and software technology to a network while retaining the compatibility of the previous technology.

Network Environments

The Digital communication network environments allow information to be transferred between Digital systems and equipment, and between Digital systems and systems and equipment developed by other manufacturers. Network communication between Digital systems is facilitated by DECnet software and hardware, which is a Digital developed network protocol that permits fast and efficient data communications between Digital's products. For local and remote communications between Digital systems and systems developed by other manufacturers, Digital provides Internet and Packetnet products.

Digital offers a complete range of network products for the basic point-to-point and multipoint communications links to complete network communication facilities, including sophisticated network hardware and software for complex applications. These products include single modems and multiple modem enclosures, single and multichannel, synchronous and asynchronous interfaces, communication multiplexers and controllers, and terminal servers.

Ethernet Local Area Networks

A local area network (LAN) is a privately owned data communication subsystem that provides high-speed communication channels between terminals, devices, and data processing equipment within a local area. The communication network can be routed internally in a building, or between buildings that are confined to a local area or site. Local area networks permit information and data processing resources to be shared locally and with remote sites through gateway and router server devices.

Digital's Ethernet local area network is supported by DECnet software and provides an economical means of communication. Systems and devices are easily connected to the network and network expansion is facilitated without interruption of the network operation. The DECnet software, through the Digital Network Architecture (DNA), defines the standards for protocols, interfaces, and communication functions. The DNA provides a flexible communication system based on communication layers. Each layer performs a specific set of functions and services and the interaction of the layers control the operation of the network.

Ethernet can be implemented on either baseband or broadband cable. With baseband, a single network cable replaces the many cables normally used for data networks and Ethernet is the only channel on the cable. Ethernet on broadband is an extension of baseband and allows video, voice, and other data communications to be transferred through one or two cables. This information can be shared with other independent channels. Some of the features provided by Ethernet are as follows:

- Simplified network design allows easy installation of new devices without interrupting communications between existing devices.
- Single or dual network cables replace many cables typically used in network applications.
- Cable segments can be added to expand networks.
- High-speed communication between nodes can be up to 10 Mbits per second.
- Remote locations have fast access to data.
- Allows special purpose peripheral devices, including high-speed printers, large-capacity storage devices, and high-resolution graphics packages to be shared by users.
- Transfers text, electronic mail, facsimile data, video and audio signals.
- Enables the connection to other manufacturers' equipment.

Figure 2-8 shows a typical Ethernet baseband network configuration. Host processors, personal computers, terminal servers, gateway routers, and repeaters connect to the Ethernet cable through the H4000 Ethernet transceivers. The communications controller in the host is connected to the H4000 transceiver through a four-twisted-pair wire cable.

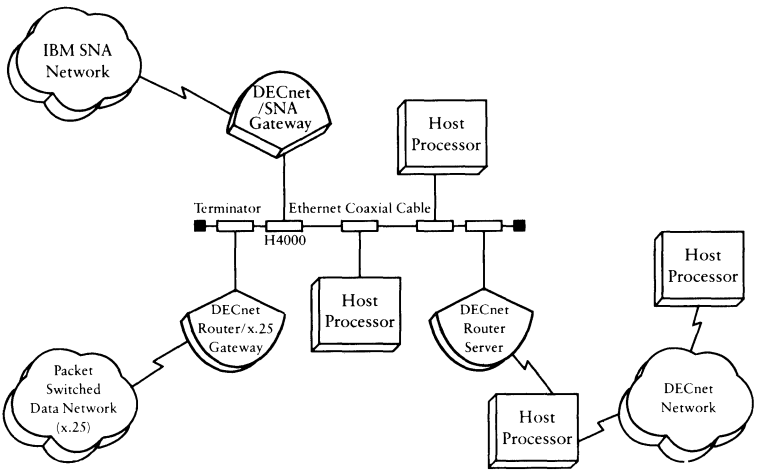


Figure 2-8 • Typical Ethernet Baseband Configuration

• ETHERNET ON BROADBAND

Ethernet broadband is a high-speed communications network that provides the same capabilities as baseband plus capabilities that allow the transfer of audio and video information. The broadband Ethernet transceiver implements DECnet Phase IV software and the same controllers are used as with baseband. Broadband networks also use the same terminal and communication servers, concentrators, DECnet routers, and gateways as the baseband network except that network repeaters are not required. Broadband operates with a single coaxial cable or with dual coaxial cables. The coaxial cables are RG-59-U type, and the same type of cable connectors, signal splitters and amplifiers that are used for cable TV can be used with the broadband equipment. Figure 2-9 shows a typical Ethernet broadband network configuration. The communications controller in the host processor connects to the DECOM transceiver through a four-twisted-pair wire cable and the DECOM transceiver connects to a tap in the broadband network cable through a broadband coaxial drop cable. The twisted-pair cable is the same type used with the H4000 transceiver on the baseband network.

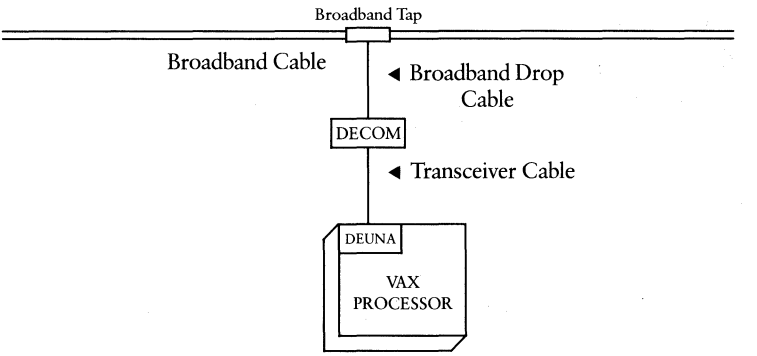


Figure 2-9 ▪ Typical Ethernet Broadband Configuration

DECOM Ethernet Broadband Transceiver–The DECOM Ethernet broadband transceiver, shown in figure 2-10, is a tabletop unit and is the physical and electrical interface to the broadband coaxial cable. The transceiver transmits signals to and receives signals from the connected systems and detects message collisions. Two types of DECOM transceivers are available, one for the single-cable broadband network and one for a dual-cable broadband network.

The single-cable transceiver (DECOM-BA) transmits 54 to 72 megahertz signals and receives 210.25 to 228.25 megahertz signals. This transceiver requires a frequency translator (DEFTR) at the headend of the network to convert the Ethernet signals from their transmit to their receive frequencies.

The dual-cable transceiver (DECOM-AA) transmits and receives the Ethernet signals at the same frequencies–from 54 to 72 megahertz.

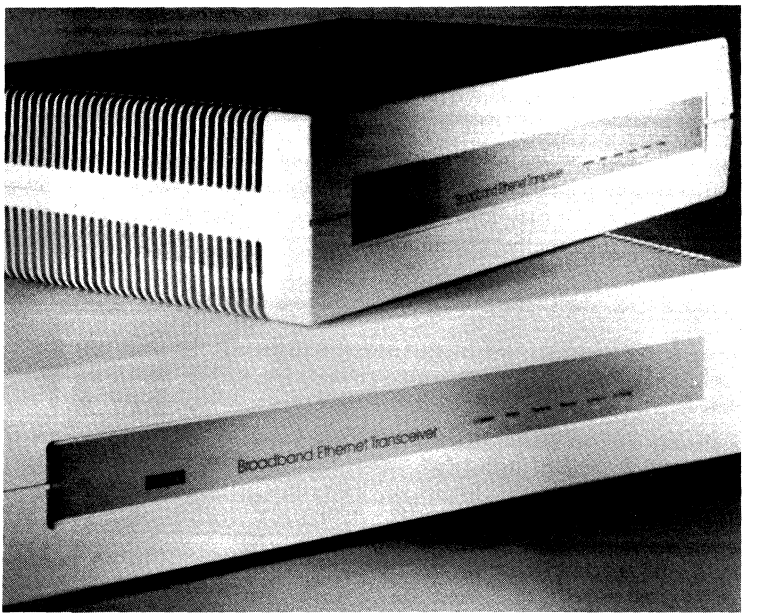
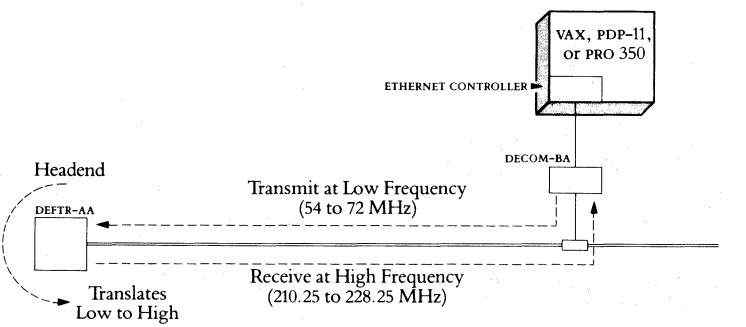


Figure 2-10 • DECOM Ethernet Broadband Transceiver

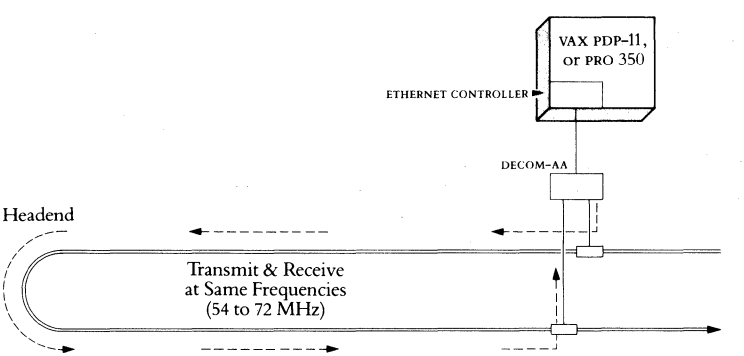
The features and benefits of the DECOM transceivers are:

- Transceiver can be located close to other Digital products.
- Channel transmissions are within assigned bandwidths, eliminating channel interference.
- Diagnostic indicator lights verify correct configurations and provide an aid for servicing the unit.
- Internal self-test and loopback capabilities allow fast diagnosis and fault isolation.
- Redundant protective circuits insure reliability and confine failures to a single transceiver, preventing loss of the entire network.
- Message collisions are reported to all transceivers to allow the Ethernet to be monitored at any transceiver location.

DEFTR Ethernet Broadband Frequency Translator—The DEFTR frequency translator is used to enable bidirectional communication transfers on a single-cable Ethernet system. It receives all Ethernet signals transmitted by the systems connected to DECOM transceivers, translates them, and retransmits the signals at the appropriate DECOM receive frequencies. The unit is installed at the headend of the network as shown in figure 2-11. It contains front panel monitors to aid in isolating both headend transceiver problems. Two DEFTR units can be connected in parallel to increase the reliability of the Ethernet communication.



Single-Cable broadband networks



Dual-Cable broadband networks

Figure 2-11 ▪ DEFTR Ethernet Broadband Frequency Translator

Some of the features of the DEFTR unit are:

- Two DEFTR translators and a switching unit are available as an option to enable information on the Ethernet to be automatically switched from one DEFTR unit to another if a failure occurs in one unit.
- Front panel monitors assist in the diagnosis and isolation of headend and transceiver failures.
- Allows constant access to the Ethernet channel.

- ETHERNET COMMUNICATION CONTROLLERS

The Ethernet communication controller (DEUNA) is an intelligent high-performance synchronous interface that mounts on the VAX processor UNIBUS and allows communication with either the baseband or broadband Ethernet network. The controller consists of two hex-height modules that are installed in the UNIBUS backplane and connect to the H4000 transceiver in the baseband network or to the DECOM transceiver in the baseband network. The controller, which is supported under DECnet Phase IV software, provides the Ethernet data link layer functions and part of the physical channel functions. The DEUNA allows communication of up to 1,023 addressable devices on the Ethernet network. Figure 2-12 shows the DEUNA controller, the H4000 transciever, and cable connections.

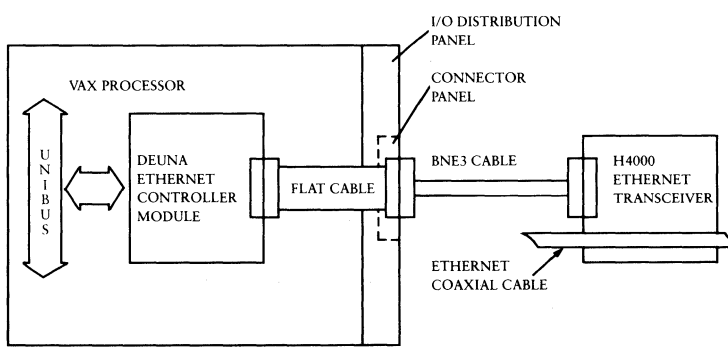


Figure 2-12 • DEUNA Controller and H4000 Transceiver Connections

- H4000 ETHERNET TRANSCEIVER

The H4000 Ethernet Transceiver, shown in figure 2-13, includes the interface circuits and hardware to connect the network nodes to the baseband Ethernet coaxial cable. It operates with the Ethernet communications controllers to transmit signals to the cable, to receive signals from the cable, and to detect message collisions that may occur. The Ethernet communications controller mounts in the UNIBUS backplane of the VAX processor and connects to the H4000 unit through a cable up to 50 meters (164 feet) long. The cable transfers the information and supplies the power to the H4000 interface circuits. The Ethernet cable segment can be up to 500 meters (1,640 feet) long or longer through the use of Ethernet repeaters.

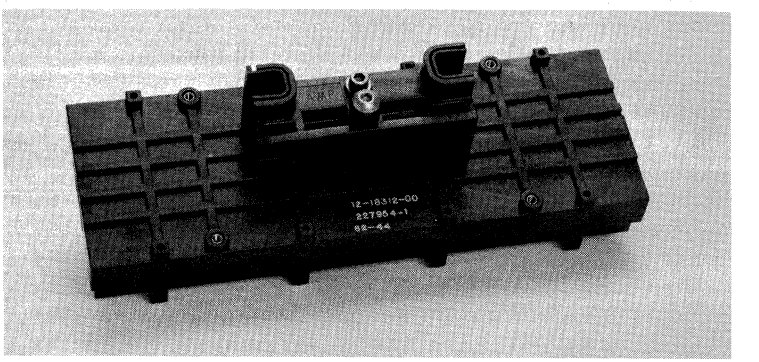


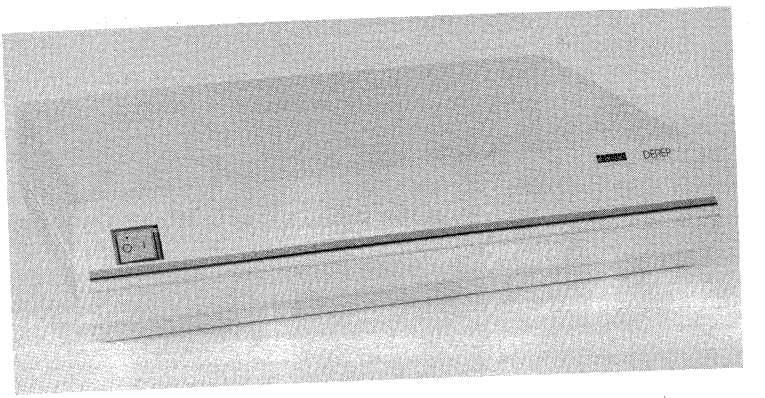
Figure 2-13 ▪ H4000 Baseband Ethernet Transceiver Unit

The H4000 transceiver contains a logic module and includes a unique cable clamp that makes contact with the internal cable conductor and holds the cable in place. A special installation kit is available to prepare the cable for installation. The cable can be installed or removed from the unit without interrupting the network operation. The H4000 unit contains redundant protective circuits and self-testing functions and provides high noise immunity to the signals transferred through the cable. A continuous message loopback feature simplifies fault isolation to reduce maintenance time and cost. The unit electrically connects to the controller through the transceiver cable that attaches to the end of the housing, as shown in figure 2-12.

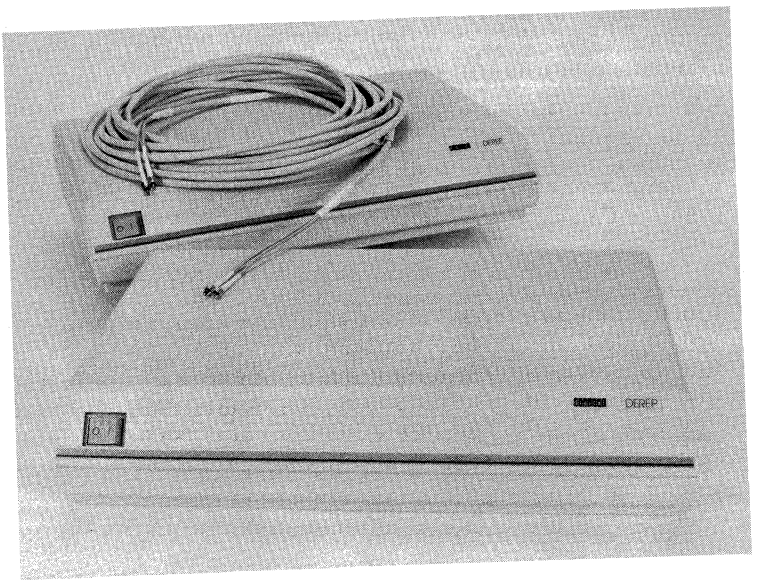
Local and Remote Ethernet Repeaters – The Ethernet repeaters (DEREP) are tabletop, stand-alone units that connect separate segments of the baseband Ethernet coaxial cables together, thereby increasing the effective length of a single cable. The repeaters connect to each of the two cables through the H4000 transceiver and extend the coaxial cable beyond the single cable length of 500 meters (1,640 feet). Up to 99 H4000 transceivers can be connected to each cable segment. The repeaters contain an internal power supply and logic to retime, amplify, and repeat the signals it receives from the H4000 transceiver. The features of the repeater unit are as follows:

- Increases the effective length of the Ethernet coaxial cable and the number of devices supported.
- Installed without interrupting network operation.
- Contains self-test features to reduce maintenance cost and increase reliability.
- Remote-repeater fiber optic cable can be installed underground or in a harsh environment.
- Detects faulty signals insuring integrity of data transmissions.
- Contains diagnostic lights to assist in network troubleshooting.
- Redundant repeaters can be installed in a network to increase reliability of transmissions.

Both local and remote repeaters are available; they connect to the Ethernet cable segments through the H4000 transceiver and transceiver cables. The local repeater connects two coaxial cable segments that are separated by a distance of up to 100 meters (328 feet); the remote repeater connects two cable segments up to a distance of 1,000 meters (3,280 feet) apart. The local and remote repeaters are shown in figure 2-14.



Local Ethernet Repeater



Remote Ethernet Repeater

Figure 2-14 ▪ Ethernet Network Repeater Units

The local repeater is a single unit that provides the transmission path betweeen the two cable segments through the H4000 transceiver. The remote repeater consists of two local repeater units that are connected by a fiber optic cable. The total distance between the Ethernet cables can be 1,000 meters (3,280 feet). A fiber-optic logic interface module is available as an option and can be added to a local repeater unit to perform as one of the remote repeater units. The network configuration of the local and remote units is shown in figure 2-15.

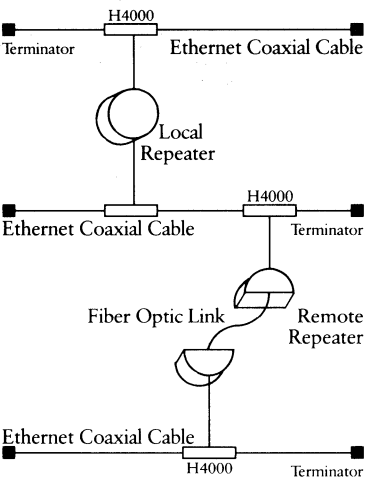


Figure 2-15 • Ethernet Network Repeater Configuration

Local Network Interconnect – The device local network interconnect (DELNI) is a low-cost, tabletop, concentrator unit that allows up to eight processors, servers, or other Ethernet-compatible devices, except terminals, to be connected together or to be connected through the DELNI and H4000 transceiver to an Ethernet cable. The unit contains a power supply and can be placed at any convenient location where ac power is available. The DELNI operates either as a stand-alone unit not connected to an Ethernet cable or in a network configuration connected to a cable. Because up to eight processors or Ethernet units can be connected, the cost of adding servers and processors to the network can be reduced using the DELNI instead of the Ethernet hardware normally required. The DELNI unit, shown in figure 2-16, provides the following features:

- Reduces the cost of multiple connections to the Ethernet coaxial cable.
- Connects Ethernet compatible devices in small networks, thereby eliminating the need for coaxial cable and H4000 transceivers.
- Contains a mode switch to allow the unit to be changed from stand-alone to connected configuration.
- Permits more than 100 devices to be connected to a single coaxial cable segment.

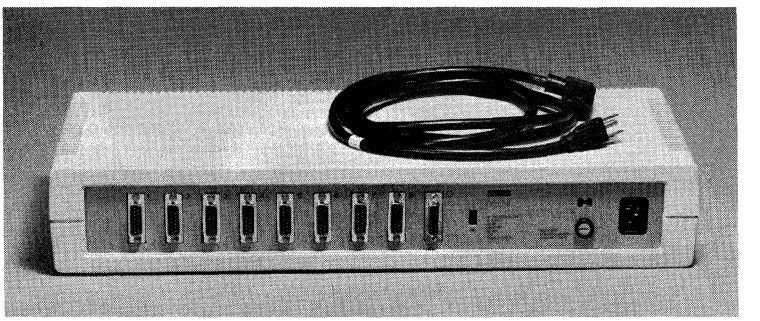


Figure 2-16 ▪ DELNI Local Network Interconnect Unit

In the stand-alone configuration, two DELNI units can be connected together to increase the total number of devices. The distance between the DELNI units can be up to 50 meters (164 feet). Figure 2-17 shows the DELNI stand-alone configuration.

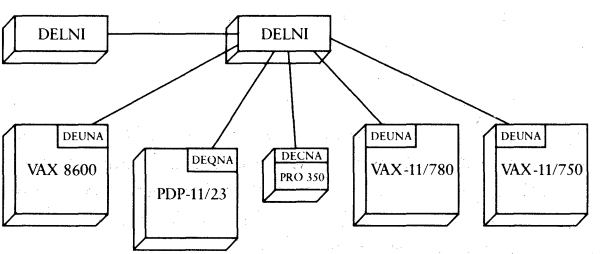


Figure 2-17 ▪ DELNI Stand-alone Configuration

In the network configuration, the DELNI connects to the H4000 transceiver through the transceiver cable and up to eight devices can be connected to the DELNI unit as shown in figure 2-18. The transceiver cable can be a direct connection to the H4000 transceiver or can be connected through the Etherjack wallplate.

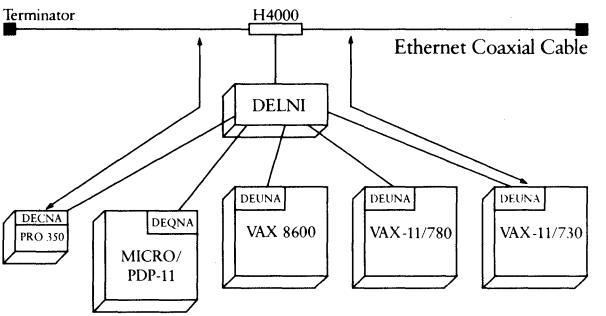


Figure 2-18 • DELNI Network Configuration

- ETHERNET COMMUNICATION SERVERS

  Ethernet communication servers are microprocessor-controlled, special-purpose units that connect terminals to an Ethernet network or connect an Ethernet network to other Ethernet and local area networks (LANs) through communication lines. All communication servers connect to the Ethernet LAN through the H4000 transceiver and cable. The Ethernet communication servers are programmable units that perform dedicated communication functions. They contain maintenance operation protocols to enable the operating and diagnostic software to be loaded into the server memory from the Ethernet host processor. In the event of a software malfunction, the server will attempt to transfer the memory information to the host processor for evaluation and will reload the software again into memory. All Ethernet communication servers are the same size and have a similar physical configuration, as shown in figure 2-19.

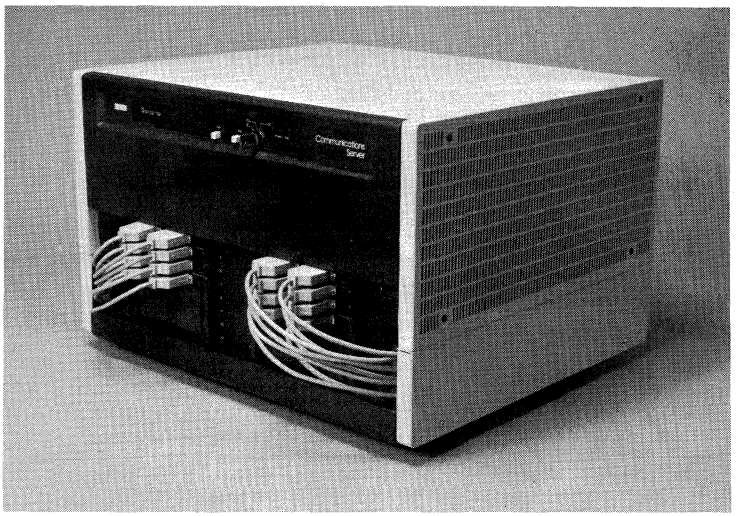


Figure 2-19 ▪ Typical Ethernet Communication Server

Four communication servers are available for operation with the Ethernet network: the Terminal Server, the DECnet Router Server, the DECnet Router/X.25 Gateway Server, and the DECnet Router/SNA Gateway Server. These units provide multiple CPU access and full modem control, and the software can be downline loaded from the host processor. Each unit contains power supplies, a PDP-11/24 processor, an Ethernet-to-UNIBUS communication controller, console/bootstrap terminators, up to one Mbyte of memory, protocol assist modules, line protocol firmware, and special function software that is optionally available. Each server is dedicated to perform a specific function; however, when communication requirements change, the server can be adapted to the new functions by replacing the line protocol firmware and the software. The features of the communication servers are as follows:

- Provides resource sharing among multiple processor systems within an Ethernet LAN
- Performs communication processing, eliminating CPU involvement
- Server operation is checked by automatic diagnostic tests performed during system startup
- Line cards can be tested online to reduce maintenance time and to minimize disruption of communications
- Changes in communication functions are easily implemented in the unit

Terminal Server–The Terminal Server provides a cost-effective and flexible means of connecting nonintelligent terminals to an Ethernet network. Each connected terminal can access any host processor operating with VAX/VMS or RSX software on a network of remote DECnet nodes. Access is through DECnet router servers connected to the same Ethernet network. Terminal access to the DECnet Router/X.25 Gateway or DECnet/SNA Gateway Server is through the VAX/VMS host system that contains the appropriate access routines. The features and benefits of the Terminal Server are as follows:

- Provides terminal access to multiple local hosts on the same Ethernet network
- Performs I/O processing functions normally performed by the host CPU
- Same Terminal Server software operates with a 16- or 32-line hardware configuration
- Ability to access multiple host processors reduces the number of terminals connected to individual host processors
- Increases reliability and redundancy by allowing terminals logically connected to a failed host processor to access other host processors on the Ethernet network

Up to 32 asynchronous terminals can be connected to the Terminal Server locally through null modem cables or remotely through modems and dial-up lines. Through simple commands, the user establishes a logical connection to a system and has access to the standard commands and utilities supported by the node. The Terminal Server supports the simultaneous operation of the terminals at full-duplex rates of from 50 to 19,200 bits per second to local Ethernet host processors that implement the local area host (LAH) protocol. The Terminal Server controls and monitors the communication from the terminal to the host processor through a single Ethernet interface resulting in a lower cost per line than with the traditional data switches. The unit allows split-speed terminal operation and provides modem control and monitoring, and automatic line-speed detection. The Terminal Server supports EIA standard RS-232-C and CCITT standard V.28 for data transmission. A typical Ethernet network configuration with terminal servers is shown in figure 2-20.

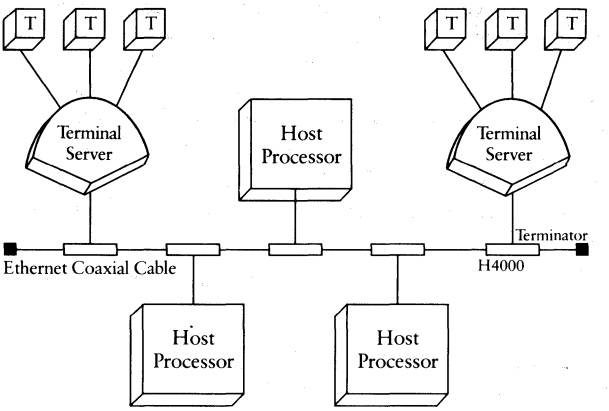


Figure 2-20 ▪ Terminal Server Network Configuration

DECnet Router Server–The DECnet Router Server performs the routing functions to permit the host VAX processors on an Ethernet network to communicate with local or remote DECnet networks. The DECnet Router implements DECnet Phase IV protocols and network management functions and can communicate with processors using DECnet Phase III or Phase IV software. Using the DECnet Router, the computer networks can consist of up to 63 areas, each with up to 1,023 nodes. The DECnet Router connects to the Ethernet through the H4000 Transceiver and receives and forwards messages between the DECnet network and Ethernet networks. Figure 2-21 shows a typical DECnet Router network configuration.

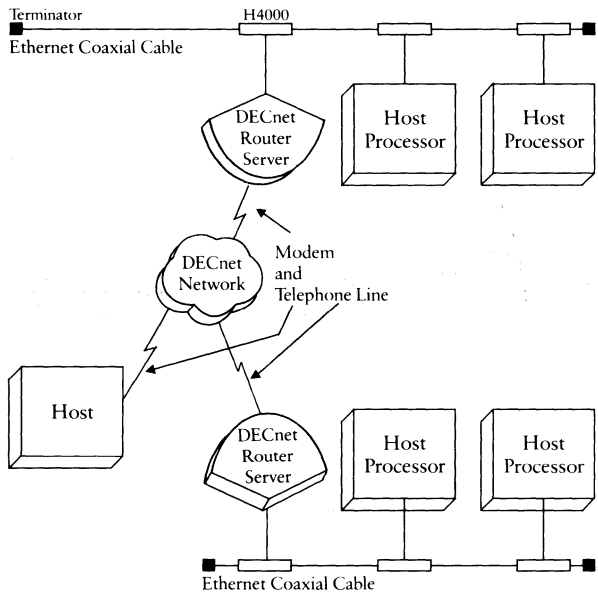


Figure 2-21 • DECnet Router Network Configuration

The DECnet Router supports the direct connection of up to eight low-speed, single-line, synchronous EIA standard RS-232-C or CCITT standard V.28 interfaces operating at full-duplex transfer rates of up to 19,200 bits per second or up to eight high-speed synchronous devices at full-duplex transfer rates of up to 56,000 bits per second. The unit optionally supports a single 500-Kbit-per-second synchronous line or a two 250-Kbit-per-second connection through a CCITT V.35 standard line interface. A DECnet Router is not required if the end nodes connected to the Ethernet network communicate only with each other. The network management functions of the router are performed by the Ethernet load host and include the control and monitoring of the router operation and the fault analysis of network problems.

DECnet Router/X.25 Gateway Server—The DECnet Router/X.25 Gateway connects DECnet nodes on an Ethernet network to DECnet nodes on an X.25 packet-switched data network (PSDN) using the data link mapping (DLM) capability of the Gateway software. The DECnet Router/X.25 Gateway connects to an Ethernet through the H4000 Transceiver and to a remote DECnet network through a modem and a telecommunication line. The unit supports six lines at data rates up to 56 Kbits per second, 16 and 32 lines at rates up to 19.2 Kbits per second, two lines at 250 Kbits per second, and one line at 500 Kbits per second. The features and benefits of the DECnet/X.25 Router are as follows:

- Allows communication between applications being performed on VAX systems and other manufacturers' systems.
- Allows an X.25-compatible PSDN to be used as the communication medium for DECnet, non-DECnet, and Digital-to-non-Digital data traffic.
- Permits communication between remote terminals and VAX host processors on the Ethernet network using the X.3, X.28, and X.29 protocols.
- Shares the PSDN access among the systems on the Ethernet network.

The unit includes the same function as the DECnet Router as well as additional functions. Host VAX processors on the Ethernet network that contains a VAX X.25/X.29 Extension Package have access to the facilities offered by the PSDN to which the router is connected. This allows remote terminals to access the host VAX processor by dialing in through the network packet assembler/disassembler (PAD), which is included in the extension package. Local terminals that are directly connected to the VAX processor or are logically connected through the Terminal Server can access other systems connected to the PSDN by dialing out through the PAD facility. The extension package also allows access to the packet level of the protocol software so that the VAX system user can design utility software for task-to-task communication between the VAX system and systems developed by other manufacturers. The Ethernet network configuration, including the router, is shown in figure 2-22.

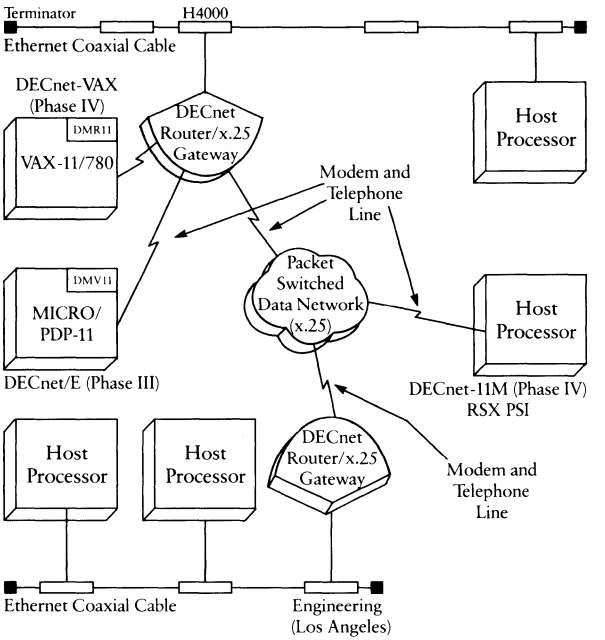


Figure 2-22 • DECnet Router/X.25 Gateway Network Configuration

DECnet/SNA Gateway Server–The DECnet/SNA Gateway server allows the Ethernet network to communicate with remote IBM® System Network Architecture (SNA). The unit connects to the Ethernet network through the H4000 Transceiver and to the SNA network through the internal modem and telecommunication line. The unit supports data transfer rates of six lines at up to 56 Kbits per second, eight lines at up to 9.6 Kbits per second, seven lines at 19.2 Kbits per second, two lines at a rate of 9.6 Kbits per second, and one line at 256 Kbits per second. The features and benefits of the DECnet/SNA Gateway are as follows:

®IBM is a registered trademark of International Business Machines Corporation.

- Provides the complementary features of both Digital and IBM network environments.
- Provides 3270-terminal emulation, remote job entry, and user application interface functions of the SNA.
- Includes gateway management software for control and maintenance.

The DECnet/SNA Gateway Server implements the SNA Gateway protocol and is used with the appropriate VAX/VMS gateway-access software. Figure 2-23 shows the network configuration using the DECnet/SNA Gateway Router.

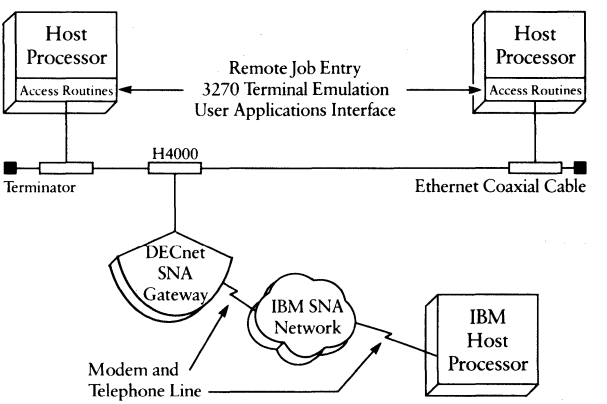


Figure 2-23 ▪ DECnet/SNA Gateway Network Configuration

UNIBUS Communication Interfaces and Devices

Digital provides a series of UNIBUS interfaces and devices that enable communication between processors and between processors and terminals. The processors and terminals can be confined to the same site or located at different sites. Communication between the remote sites is facilitated through the use of private or public telecommunication lines. The interconnection of local and remote equipment is through a network configuration. A network is a flexible configuration of interconnecting terminals, computers and other intelligent devices that are defined as nodes. The communication hardware and software allow data transmission between Digital's processors and equipment, and between Digital's processors and other manufacturers' equipment. The transmission methods or protocols that are required depend on the type of processors and devices that are communicating with each other.

Asynchronous transmission uses a character orienter protocol and is primarily intended for intermittent low-speed data transmissions between terminals and their host computer. The transmission is controlled by the start and stop elements of the character and the time intervals between character transmission is variable.

Synchronous transmission is used primarily for high-speed data transfers between computers. It uses a block-oriented protocol and the data is transferred in one interval as a group or block of characters. The data characters and bits are transferred at a fixed rate and the transmitter and receiver are synchronized.

The digital signals that are transferred through most telephone lines are converted to analog signals by a modem. The modem also reconverts the analog signals to digital signals upon reception. Computers and devices connect to the modems through standard interfaces that provide the proper signal levels and characteristics. The interfaces are designed according to standards from the Electronic Industries Association (EIA) and from the International Consultative Committee on Telephony (CCITT). Some of these standards for telephone equipment in data transmission are EIA RS-232-C/CCITT V.28, EIA RS-422/CCITT V.11, EIA RS-423/V.10 and CCITT V.35. All Digital communication interfaces that conform to the industry standards can be used to connect Digital systems to private branch exchanges (PBXs) through modems.

- UNIBUS ASYNCHRONOUS INTERFACES

Single and multiline asynchronous interfaces are provided for communication between the VAX processors and terminals. The interface selection is based on the number of lines, transmission speed and characteristics, and the modem control requirements. These interfaces allow point-to-point communication and multipoint communication between the nodes on the network. The interfaces include the following:

- DMF31–A multifunction communication controller that provides an eight-line asynchronous multiplexer, a single-line asynchronous interface, and a general purpose parallel interface
- DMX32–A 24-line asynchronous multiplexer with DMA capabilities on transmit lines and program-selectable split-speed transmit and receive rates
- DZ11–An eight-line asynchronous multiplexer with program selectable transmit and receive rates
- DHU11–A 16-line asynchronous multiplexer with DMA capabilities and program-selectable transmit and receive rates

DMF32 Multifunction Communication Controller–The DMF32, shown in figure 2-24, is an intelligent multifunction communication controller that permits modems and terminals to be connected to the UNIBUS. The DMF32 consists of an eight-line asynchronous multiplexer, a single-line asynchronous interface, and a general purpose parallel interface. It is supported by VMS operating software, DECnet-VAX networking software, and VAX PSI communications software. Some of the features and benefits of the controller are:

- Allows concurrent operation of the three controllers on one module
- Provides two asynchronous channels with full modem control and individually selectable transmit and receive baud rates
- Enables six local terminal connections
- Provides a parallel interface for DMA or SILO mode operation with a user device or intelligent DMA line printer controller
- Performs control of I/O transfer operation, eliminating the processor intervention

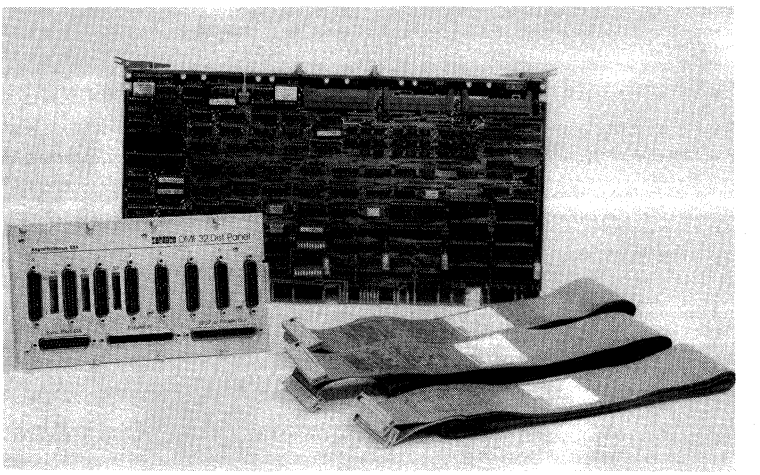


Figure 2-24 ▪ DMF32 Multifunction Communication Controller

The eight-line asynchronous multiplexer provides full-duplex operation at speeds up to 19,200 bits per second. The data transmission can be in DMA mode or first-in, first-out SILO mode. Two of the lines provide full modem control and split transmission- and reception-speed capabilities. Six lines are dedicated to local terminals and no modem control is included. The single-line synchronous interface operates at speeds up to 19,200 bits per second in the DMA mode and provides full modem control and support for both bit- and byte-oriented protocols. The general purpose parallel interface operates with either a line printer in DMA mode or with a user device in DMA or SILO mode. The DMF32 is compatible with Digital's family of modem devices and mounts in one hex slot of the UNIBUS backplane. Figure 2-25 shows the configuration of the controller and the I/O connections.

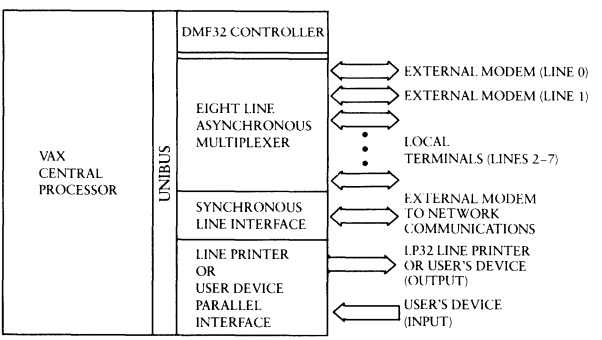


Figure 2-25 • DMF32 Multifunction Controller Configuration

DMZ32 24-Line Asynchronous Multiplexer—The DMZ32, shown in figure 2-26, is a 24-line asynchronous multiplexer that provides full modem control and direct memory access (DMA) capability on the transmit lines. It enables the interconnection of remote terminals that conform to EIA RS-232-C/CCITT V.28 to the host VAX processor through a modem. The DMZ32 operates at full- or half-duplex transmission speeds up to 19,200 bits per second. The operating speed and line characteristics are selected by the program. Split-speed transmit and receive rates are supported on each of the 24 lines. The features and benefits of the DMZ32 are:

- Provides full modem control and split-speed capabilities on 24 lines
- Distribution panel can be mounted separately from the host processor in any 48.26-centimeter (19.0-inch) rack of cabinet
- Transmission rates are program-selectable

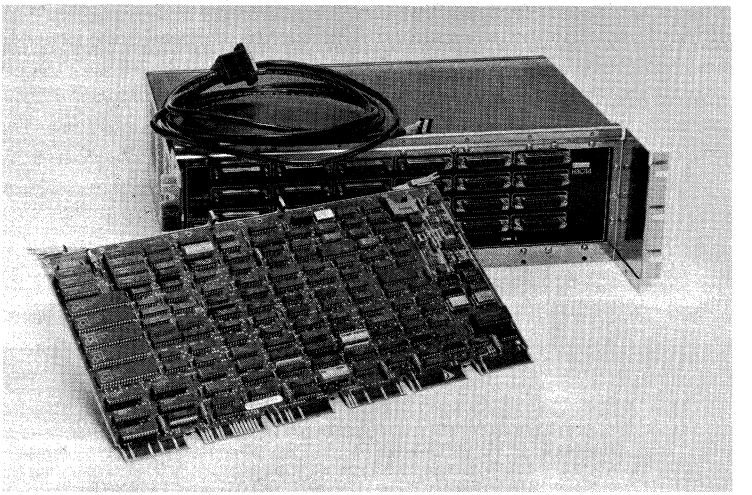


Figure 2-26 ▪ DMZ32 24-line Asynchronous Multiplexer

The distribution panel connects to the single UNIBUS interface module through a T1 cable that can be up to 1,524 meters (5,000 feet) long. The panel can be mounted in the H9646 modem cabinet or any 48.26-centimeter (19.0-inch) cabinet. The modem cabinet allows a combination of modem devices, including the DF100 series of modem enclosures and the DMZ32 distribution panel, to be contained in the same cabinet. Figure 2-27 shows the configuration and cables associated with the DMZ32 multiplexer.

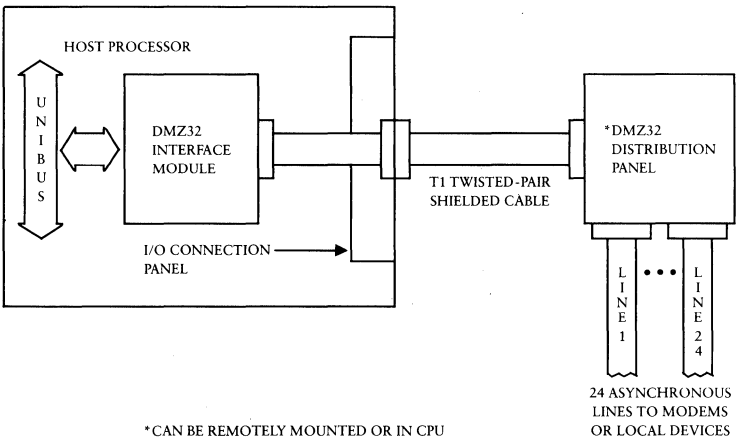


Figure 2-27 ▪ DMZ32 24-line Asynchronous Multiplexer Configuration

DZ11 Eight-line Asynchronous Multiplexer–The DZ11 option, shown in figure 2-28, is a low-cost, eight-line, asynchronous multiplexer that enables local and remote communication between VAX processors, other Digital processors, and terminals. It provides communication for both RS-232-C/ CCITT V.28 standard lines and 20-milliampere lines. The DZ11 provides buffering for full-duplex operation at program-selectable speeds from 50 to 9,600 bits per second and limited modem control. Some of the features of the option are:

- Low-cost, eight-line multiplexer
- Program-selectable transfer rates for each channel
- Compatible with Digital modems and Bell 100 and 200 series modems

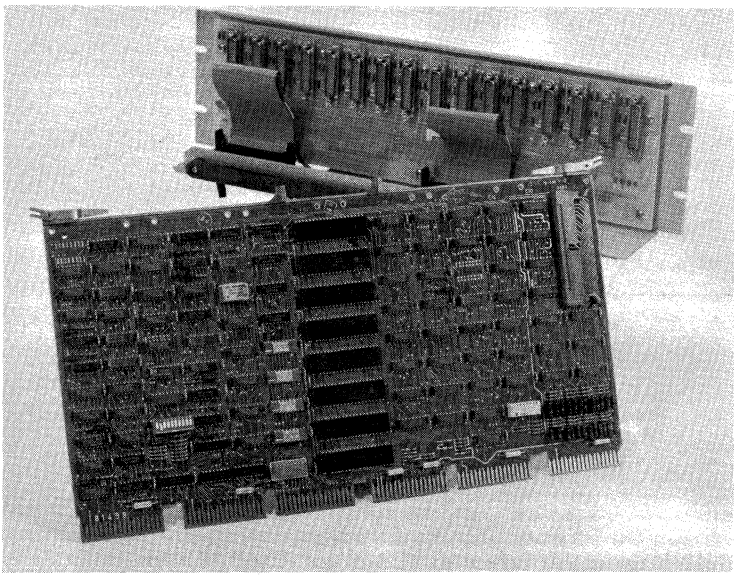


Figure 2-28 ▪ DZ11 Eight-line Asynchronous Multiplexer

The DZ11 consists of a hex-height module that mounts in the UNIBUS backplane, an interconnecting cable, and an I/O connection panel with 16 connectors for external lines. Figure 2-29 shows the DZ11 hardware configuration.

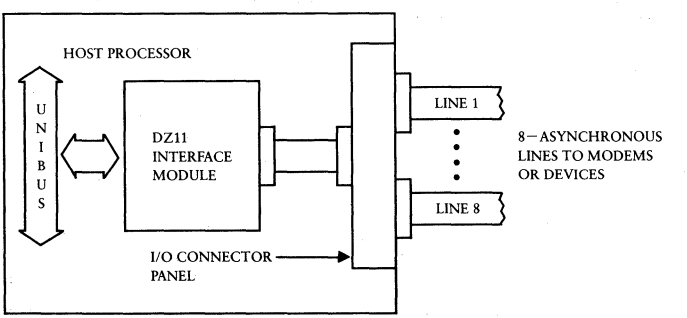


Figure 2-29 ▪ DZ11 Eight-Line Asynchronous Multiplexer Configuration

DHU11 16-Line Asynchronous Multiplexer–The DHU11, shown in figure 2-30, is a 16-line asynchronous multiplexer that allows direct-memory-access transfers for the devices connected to the lines. It can be used for local or remote communication between Digital's computers and intelligent terminals through EIA RS-232-C/CCITT V.28, or EIA RS-423-A/CCITT V.10 standard interfaces. The DHU11 provides half- or full-duplex operation at program selectable speeds of up 38,400 bits per second. Full modem control and split-speed transfer and receive rates are included on all lines. The features and benefits of the DHU11 option are the following:

- Allows DMA transfers with devices on all lines
- Provides full modem control on all lines
- Program-selectable transfer rates up to 38,400 bits per second

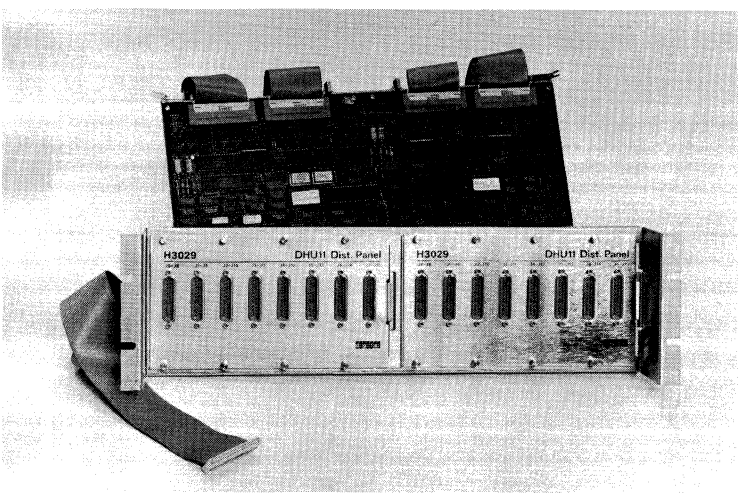


Figure 2-30 • DHU11 16-line Asynchronous Muliplexer

The DHU11 option consists of a hex-height module that mounts in a slot on the UNIBUS backplane, the internal cables and the I/O connection panel with 16 mounted connectors. The configuration of the DHU11 hardware is shown in figure 2-31.

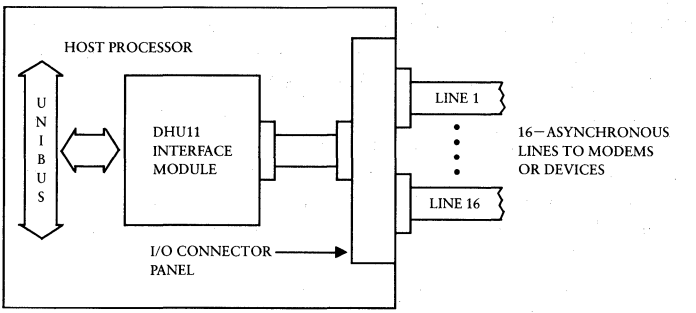


Figure 2-31 ▪ DHU11 16-line Asynchronous Multiplexer Configuration

- SYNCHRONOUS COMMUNICATION DEVICES

In addition to the DEUNA Ethernet communications controller previously described in this chapter, Digital provides a complete series of synchronous interfaces that permit high-speed data transfers between local and remote processors and intelligent terminals. These options provide point-to-point and multipoint communication with the nodes that operate with the Digital Data Communications Message Protocol (DDCMP) and other bit- and byte-oriented protocols for fast and reliable data transfer. The synchronous communication devices included are:

- DMP11—A high-performance, microprocessor-controlled, single-line synchronous interface for DMA transfers using DDCMP byte-oriented protocols in DECnet point-to-point and multipoint configurations
- DMR11—A high-performance, microprocessor-controlled, single-line synchronous interface for DMA transfers using DDCMP byte-oriented protocols in DECnet point-to-point configurations
- DUP11—A high-performance, single-line, programmable synchronous interface for byte- or bit-oriented protocols
- KMS11-B—An eight-line, microprocessor-controlled programmable synchronous controller for DMA transfers using Packnet System Interface (PSI) network software, bit- or byte-oriented protocol, and X.25 protocol
- KMS11-P—A single-line, microprocessor-controlled, programmable synchronous communication controller for DMA transfers using Packnet System Interface (PSI) network software and X.25 protocol

DMP11 Single-line Synchronous Controller—The DMP11, shown in figure 2-32, is a single-line, high-performance, microprocessor-controlled, synchronous controller that allows local or remote connection between PDP-11 and VAX processors and other computer systems. The DMP11 provides multi-CPU access that enables a message to be sent directly to one or more processors without routing through another processor. The communication is through EIA RS-232-C/CCITT V.28, CCITT V.35, EIA RS-422/CCITT V.11, EIA RS-449, or EIA RS-423/CCITT V.10 standard interfaces. The DDCMP is implemented in the controller microcode to insure maximum thoughput and minimum processor intervention. The DMP11 allows DMA transfers in DECnet point-to-point or multipoint configurations. It operates at a transmission speed of one Mbyte per second for half-duplex transfers and performs half- and full-duplex transfers for all other speeds. The DMP11 option includes full modem control. The features and benefits of the interface are:

- Point-to-point or multipoint operation
- Supports up to 32 tributaries in multipoint configurations
- Intelligent controller and hardware-implemented DDCMP reduces host processor hardware and software intervention
- Communicates with any other DDCMP devices
- Supports DMA data transfers and provides full modem control with many standard interface lines

Depending on the operating system and layered software, the DMP11 can support a maximum of 32 tributaries. In multipoint configurations, these tributaries can be DMP11, DMV11, or DMF32 interfaces. In point-to-point configurations, the DMP11 can communicate with any other synchronous interface that implements the DDCMP Version 3.1 or 4.0. The DMP11 option consists of two hex-height modules that mount in the UNIBUS backplane, I/O connector panel inserts, and an interconnecting cable. Figure 2-33 shows the hardware configuration of the controller.

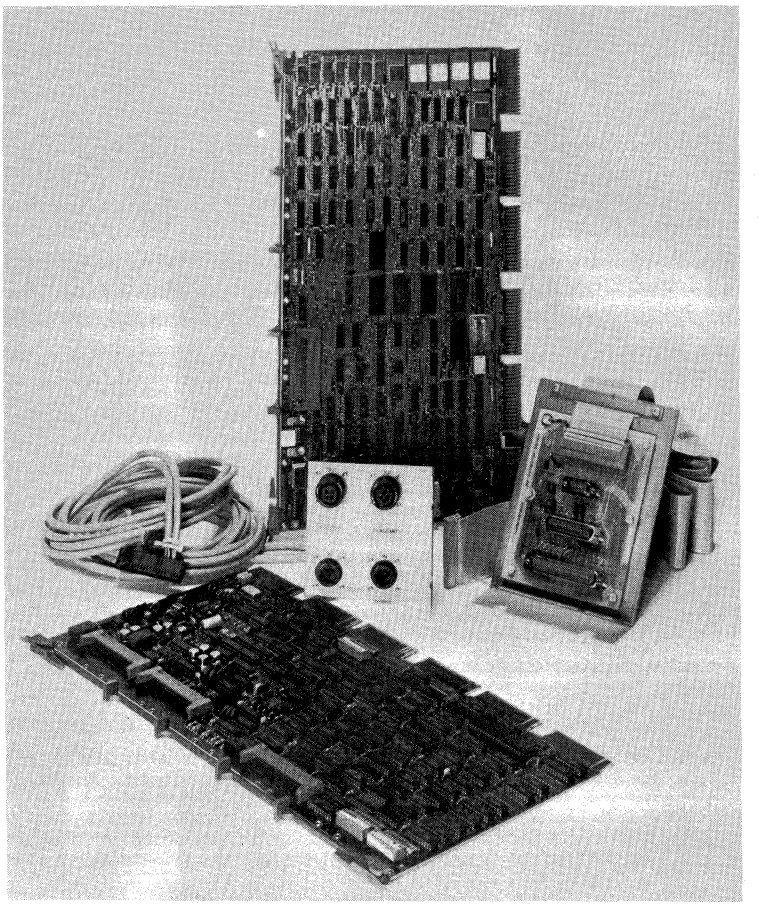


Figure 2-32 ▪ DMP11 Single-Line Synchronous Multipoint Synchronous Controller

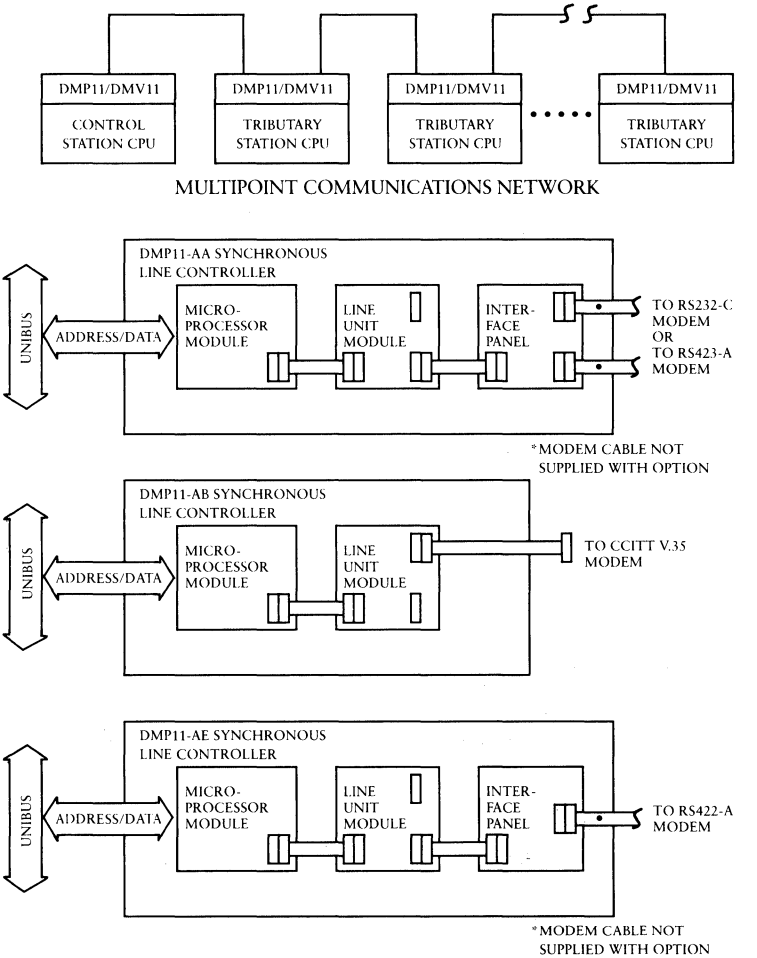


Figure 2-33 • DMP11 Synchronous Controller Configurations

DMR11 Single-line Synchronous Interface – The DMR11, shown in figure 2-34, is a high-performance, microprocessor-controlled, single-line, synchronous interface that allows local and remote connections between VAX systems and other Digital systems. The communication is through EIA RS-232-C/CCITT V.28, CCITT V.35, EIA RS-422/CCITT V.24, EIA RS-422/RS-449, or EIA RS-423/CCITT V.24 standard interfaces. The DMR11 implements DDCMP protocols in hardware and supports direct memory access data transfers and DECnet point-to-point configurations. It provides half- and full-duplex operation at transmission speeds of 1 Mbyte per second and includes full modem control. The features and benefits of the interface are as follows:

- Intelligent controller and hardware-implemented DDCMP reduces host processor hardware and software intervention
- Communicates with any other DDCMP device
- Supports DMA data transfers and provides full modem control with many standard interface lines

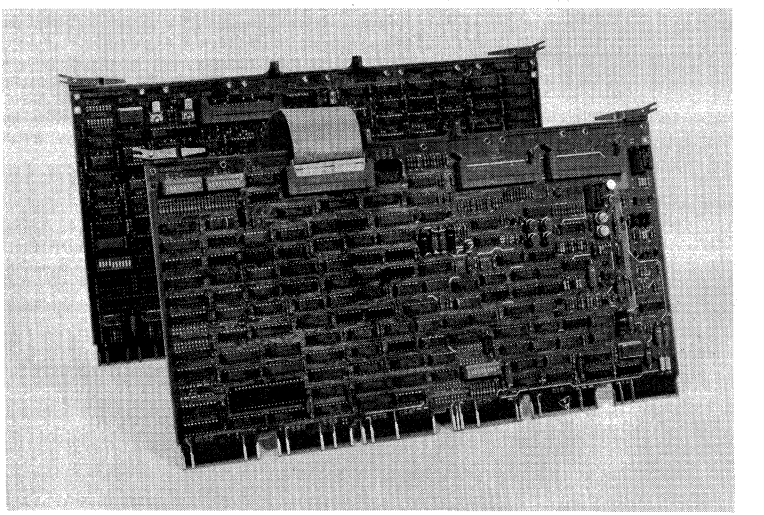


Figure 2-34 ▪ DMR11 Single-Line Synchronous Interface

The DMR11 option consists of two hex-height modules that mount in the UNIBUS backplane, an I/O connector panel insert, and an interconnecting cable. The DMR11 option is available with different interfaces, depending on the line requirements of the modem. It is compatible with Digital's family of modems and with Bell 200 series modems, Bell 500a L1/5 modem or the equivalent. Figure 2-35 shows the hardware configuration of the interface.

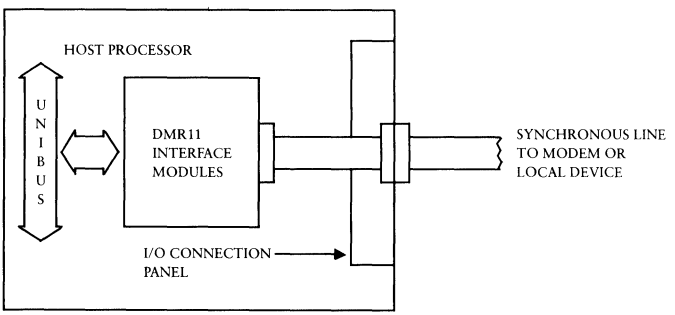


Figure 2-35 • DMR11 Synchronous Interface Configuration

DUP11 Single-line Synchronous Programmable Interface – The DUP11, shown in figure 2-36, is a high-performance, single-line, programmable interface that allows communication between PDP-11 and VAX systems and other computer systems. Communication is through EIA RS-232-C/ CCITT V.28 standard interfaces and provides half- or full-duplex operation at transfer rates of up to 9,600 bits per second. The unit provides full modem control and is programmable for byte-oriented Digital data communications message protocol (DDCMP), byte-oriented binary synchronous communications protocol, bit-oriented synchronous data link control (SDLC), or bit-oriented high-level data link control (HDLC). The DMP11 interface can be used for remote batch and remote job entry applications and is compatible with Digital's family of modems or Bell 200-series modems and their equivalents. Some of the features and benefits of the DUP11 follow:

- Programmable for bit- or byte-oriented protocols
- Provides full-modem control
- Double character-buffered receive and transmit
- Provides CRC check for DDCMP protocols

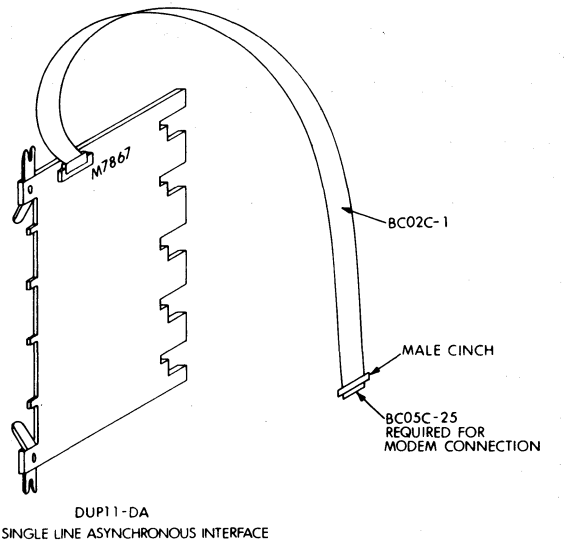


Figure 2-36 ▪ DUP11 Single-line Synchronous Programmable Interface

The DUP11 consists of one hex-height module that mounts in the UNIBUS backplane, an I/O connector panel insert, and an interconnecting cable. Figure 2-37 shows the hardware configuration of the interface.

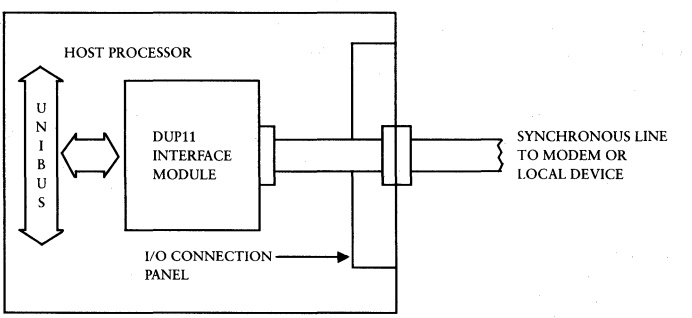


Figure 2-37 ▪ DUP11 Synchronous Programmable Interface Configuration

KMS11-B Eight-line Synchronous Programmable Controller – The KMS11-B controller, shown in figure 2-38, is an eight-line programmable communications processor that operates with the host processor and performs many of the communication functions normally performed by the processor. The KMS11 includes a 4-Kword RAM that can be programmed to perform many dedicated communication operations between PDP-11 and VAX systems and other systems. The communication is through EIA RS-232-C/CCITT V.28, MIL-188-144 unbalanced, or CCITT V.35 standard interfaces. The KMS11-B provides half- or full-duplex operation at transfer rates of up to 56,000 bits per second with full modem control. The KMS11-B supports DMA transfers, the VAX PSI software package and the CCITT standard X.25, HDLC, bit-synchronous and byte-synchronous protocols. The CCITT X.25 link-level software is currently warranted for four lines at 56,000 bits per second or eight lines at 19,200 bits per second. Some of the features of the KMS11-B follow:

- Contains a high-speed LSI microprocessor and 4-Kbyte programmable RAM for user applications programs
- Suitable for customer-developed device interface requirements
- Provides DMA data transfers for information on all eight lines
- Separate line units available for specific line requirements

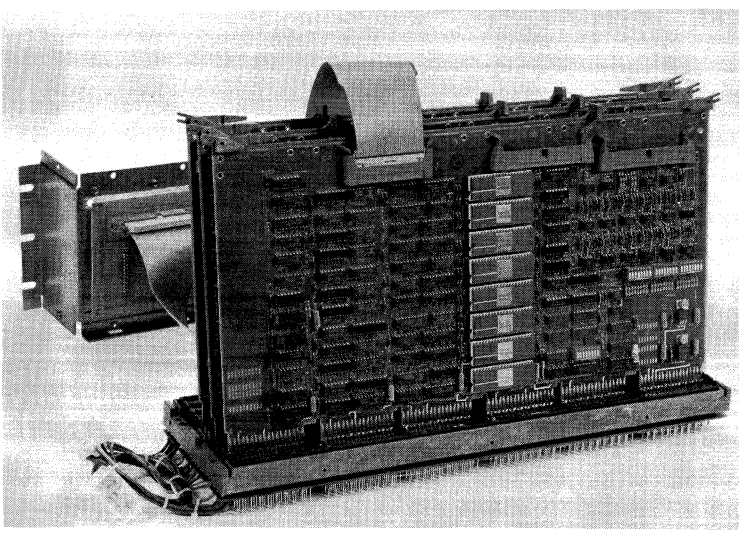


Figure 2-38 • KMS11-B Eight-line Synchronous Programmable Controller

A software tools package is available to aid in the development and debugging of user programs for special communication applications. The software tools include utility programs to load the RAM from a file and programs that enable the programmer to examine and modify the contents of internal registers and memory.

The KMS11-B consists of three hex-height modules: the microprocessor module and memory, an eight-line modem control module, and an eight-line multiplexer terminator module. The modules can be installed in an existing UNIBUS backplane or in the DD11-DK UNIBUS backplane that can be included with the option. It also includes an eight-line I/O connection panel unit and internal cables. The hardware configuration of the KMS11-B is shown in figure 2-39.

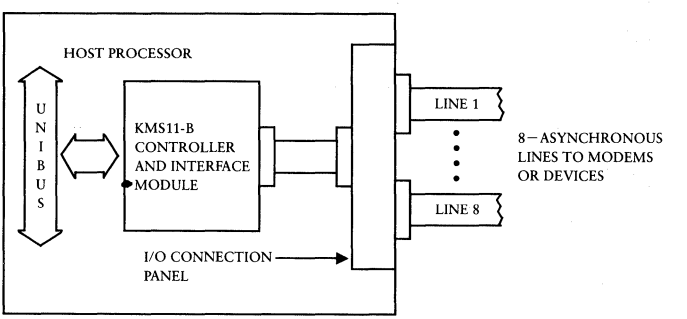


Figure 2-39 ▪ KMS11-B Eight-Line Synchronous Programmable Controller Configuration

KMS11-P Single-line Programmable Controller—The KMS11-P, shown in Figure 2-40, is a single-line programmable communications processor that operates with the host processor and performs many of the communication functions normally performed by the processor. The KMS11-P is similar to the KMS11-B and includes a 4-Kword RAM that can be programmed to perform many dedicated communication operations between PDP-11 and VAX systems and other systems. The communication is through EIA RS-232-C/CCITT V.28, EIA RS-423-A/CCITT V.10, CCITT V.35, or EIA RS-422-A/CCITT V.11 standard interfaces. The KMS11-P allows half- or full-duplex operation at a maximum transfer rate of 64,000 bits per second with full modem control. The KMS11-P supports DMA transfers, the VAX PSI software package and the CCITT standard X.25 protocol. Some of the features of the KMS11-P follow:

- Contains a high-speed LSI microprocessor and 4-Kbyte programmable RAM for user applications programs
- Suitable for customer-developed device interface requirements
- Provides DMA data transfers
- Separate line units available for specific line requirements

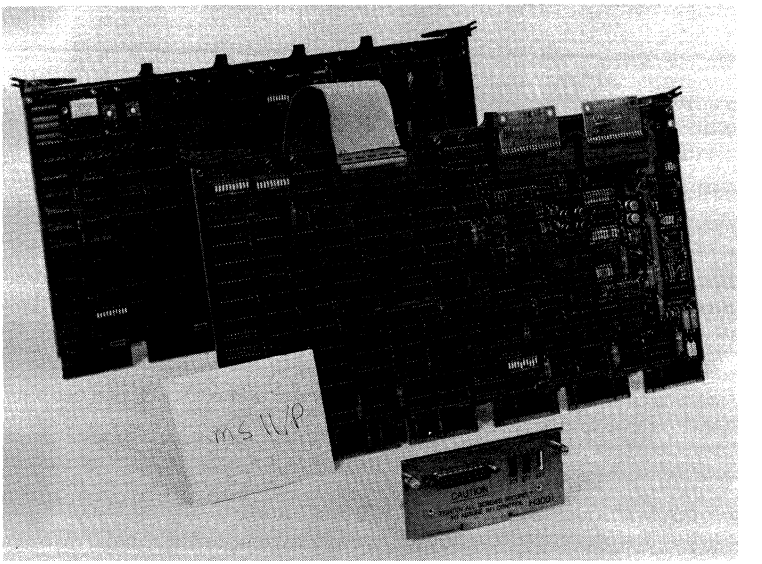


Figure 2-40 • KMS11-P Single-line Synchronous Programmable Controller

The KMS11-P consists of two hex-height modules: the microprocessor module and memory, and a line terminator module. The modules can be installed in an existing UNIBUS backplane. It also includes an I/O connection panel insert and an internal cable. The hardware configuration of the KMS11-P is shown in figure 2-41.

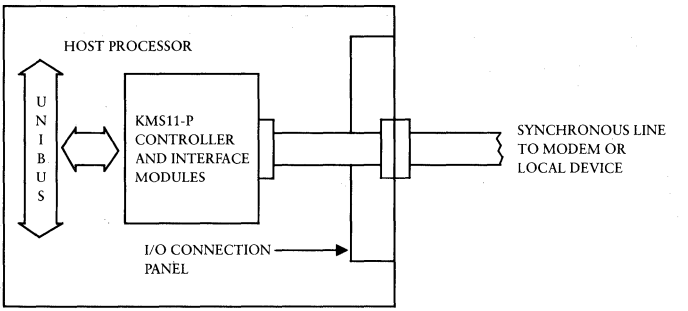


Figure 2-41 ▪ KMS11-P Single-line Synchronous Programmable Controller Configuration

▪ TERMINAL CONCENTRATORS AND MULTIPLEXERS

The DMF series of intelligent communication processors and the DZS11 terminal concentrator and statistical multiplexer provide the communication links to increase the number of terminals in a local network configuration and to increase the efficiency of the system's remote communications facilities.

DZS11 Terminal Concentrator and Multiplexer—The DZS11 is a terminal concentrator and statistical multiplexer. The DZS11 receives low-speed asynchronous communication from a maximum of eight VT100-AA or VT100-AB video terminals and transfers the information to the host processor through a synchronous line and interface. The information can be from local VT100 terminals or from remote terminals through modems. The maximum transfer rate from the UNIBUS to the terminals through the DMZ11 unit is 9,600 bits per second. The DZS11 option allows up to eight local or remote VT100 terminals to be connected to the statistical multiplexer through a 19,200-bit-per-second synchronous link and divides the communication line capacity according to the user's needs. The asynchronous lines to the terminals conform to EIA RS-232-C/CCITT V.28 standards. The features of the DMS11 option are:

- Increases the communication speed to local and remote terminals from the host processor
- Decreases the connecting lines from terminals to the host processor
- Dynamically distributes the communication information according to the user requirements
- Allows two terminal concentrators to be connected to the DZS11 unit

The DZS11 option consists of a terminal concentrator module and I/O connector panel that mount on the VT100 terminal and a hex-height statistical multiplexer interface module that mounts in the UNIBUS backplane. Up to seven VT100 terminals can be connected to the terminal concentrator module, which mounts into the backplane of a single VT100 terminal. A connector panel at the rear of the VT100 transfers the information from the attached terminals and routes the information through the terminal concentrator and to the synchronous interface installed in the host system UNIBUS backplane. The DZS11 synchronous line consists of a hex-height module that mounts into the UNIBUS backplane, an I/O connection panel insert, an I/O connection panel and an internal connecting cable. The cable connectors for the cable that connects the host processor to the VT100 terminal are supplied. Figure 2-42 shows some of the possible DZS11 configurations.

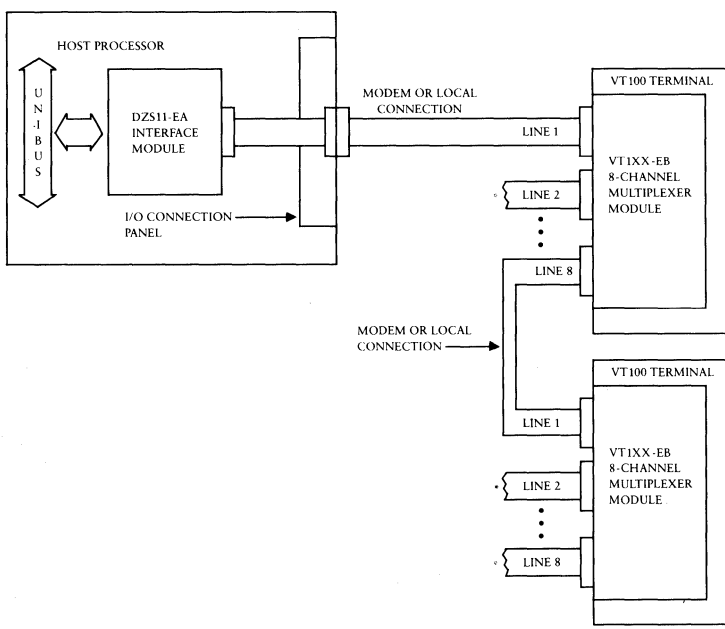


Figure 2-42 ▪ DZS11 Terminal Concentrator/Multiplexer Network Configuration

- DMF SERIES OF STATISTICAL MULTIPLEXERS

The DMF series of statistical multiplexers are intelligent communication processors that concentrate data from multiple low-speed communication lines into a single high-speed line. The DMF units support DMA transfers, asynchronous or synchronous data communication, optional integral modems, and expansion of from 4 to 16 lines. The modem transfer rates are 4,800 and 9,600 bits per second. The unit includes automatic speed detection, speed and flow control conversion, local and remote echo, fly-back character control, and dialup support on EIA standard RS-232-C/CCITT V.28 lines. A switched and contention feature allows a logical connection to be established between a user's terminal and other compatible devices connected to the ends of the network. Asynchronous speeds are from 50 to 9,600 bits per second with autobaud above 150 bits per second. The concentrated link speeds are from 1200 to 19,200 bits per second. The synchronous channels include 16-bit error detection with automatic retransmit when an error is detected. Autodial is supported on the EIA lines. The DMF unit is shown in figure 2-43. Some of the unit features are:

- Increases the communication speed to local and remote terminals from the host processor
- Decreases the connecting lines from terminals to the host processor
- Dynamically distributes the communication information according to the user requirements
- Provides extensive management and control
- Easy expansion through the addition of line modules and integral modems

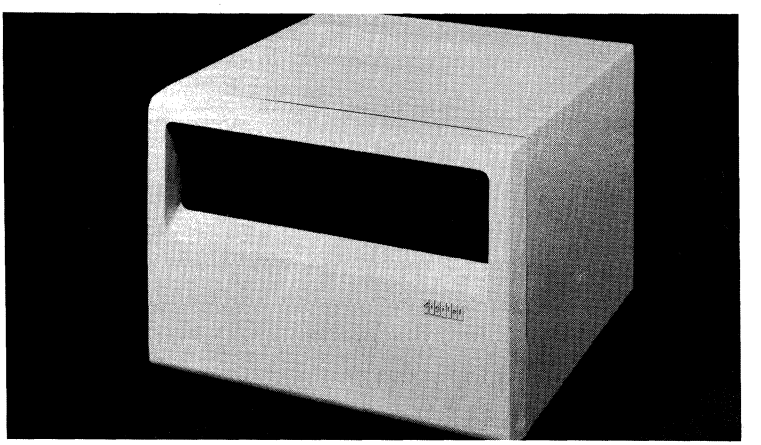


Figure 2-43 ▪ DMF Statistical Multiplexer Unit

The DMF series of multiplexers are separate units that can be placed on a tabletop or mounted into a cabinet 48.26 centimeters (19.0 inches) wide. The DMF units are available with 4, 8, 12, or 16 data channels and include a power supply and communication processor. The communication lines connect to the rear of the unit and the front cover provides access to the communication line modules. A hex-height synchronous parallel interface module that mounts into the UNIBUS backplane, an I/O connection panel, and internal cable are supplied with the option and provide the connection from the processor to the DMF unit. Some of the possible DMF multiplexer hardware configurations are shown in figure 2-44.

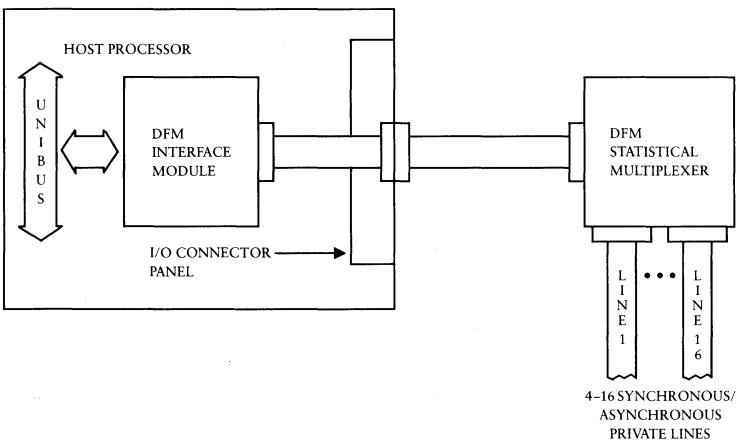


Figure 2-44 • DMF Statistical Multiplexer Configuration

PCL11-B Parallel Communications Link—The PCL11-B unit, shown in figure 2-45, is a parallel communications interface that allows multipoint connection of up to 16 processors in a local distributed network. The processors can send or receive messages or blocks of data to or from any other processors in the network. Communication occurs in a DMA block transfer mode through the time division multiplexed (TDM), 16-bit differential, parallel bus. The TDM bus allows all 16 processors to communicate concurrently using DMA transfers. The PCL11-B provides full-duplex operation at transfer rates of up to 1 Mbyte per second. The features of the PCL11-B are as follows:

- Provides efficient parallel DMA communication between up to 16 VAX processors in a local area
- High system availability and data transfer rates
- Adjustable TDM bandwidth allocations among CPU nodes

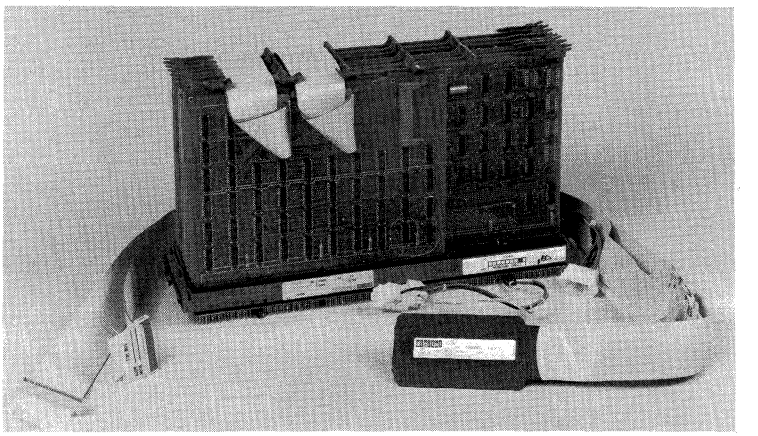


Figure 2-45 ▪ PCL11-B Parallel Communications Interface

The PCL11-B unit consists of four hex-height modules and one quad-height module that mount in a DD11-DK double system unit backplane supplied with the option. The unit connects to the I/O connection panel through an input and output cable, which is also supplied. External BC17U and BC17C cables are included. The maximum cable length is 91.5 meters (300 feet). Figure 2-46 shows a typical network configuration consisting of three computers and the PCL11-B communications interface.

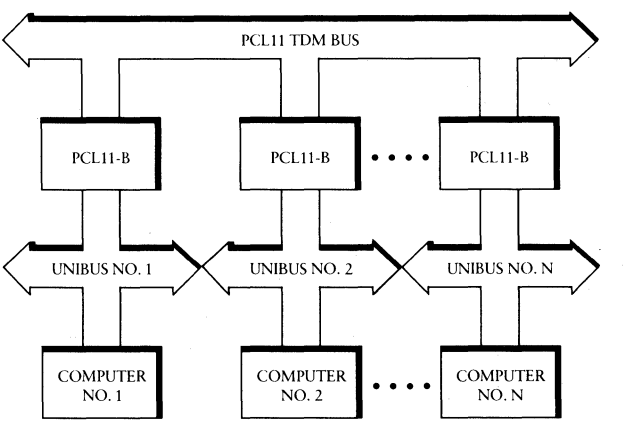


Figure 2-46 ▪ PCL11-B Parallel Communications Interface Network Configuration

DT07-xx UNIBUS Interface Switch Series – The DT07-xx UNIBUS interface switch is available as a series of options that permits UNIBUS devices to be shared by up to four PDP-11 or VAX processors. The UNIBUS section and associated devices are switched between the processors to provide high-data availability and the dynamic sharing of the data stored on devices. The DT07 is provided in three different configurations depending on the number of processors to be implemented and on mounting configuration requirements. The switching may be performed manually or under control of the program. The interface contains a timer that disconnects the UNIBUS when the controlling CPU has halted or the program is invalid and reconnects the backup processor. The features of the DT07 follow:

- Provides high data availability and sharing of data resources by up to four processors
- Switching performed manually or under control of the software
- DT07-xx options tailored to specific applications and expandable for future applications

The DT07-xx options are supplied in many different configurations. The basic option consists of a hex-height UNIBUS repeater module that mounts in a separate UNIBUS backplane and connects to the UNIBUS cables from the processor. The DT07-xx can be supplied in a mounting box with UNIBUS backplane, UNIBUS repeater and terminator modules, switching panels, and other configurations, depending the number of attached processors and the system mounting requirements. Figure 2-47 shows a typical DT07 switching configuration.

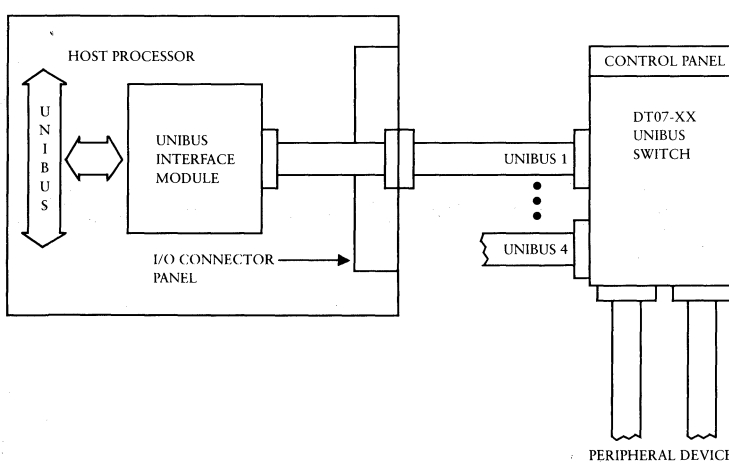


Figure 2-47 • DT07 UNIBUS Switch Configuration

FEPCM Front-End Communication Processor—The FEPCM option, shown in figure 2-48, is a powerful user-programmable, front-end PDP-11/23-PLUS or PDP-11/24 processor that can perform sophisticated control functions for many communications applications, thereby off-loading the communications handling from the host VAX processor. Because a PDP-11 processor is used, many different types of communication devices and interfaces developed for the PDP-11 are available. Through the use of the PDP-11 instruction set, a variety of existing software programs for general purpose communications applications are also available and special-purpose applications can be produced using the PDP-11 program development software. Unlike other communication processors, the FEPCM can perform the communication functions using disk storage devices, printers, and other types of interfaces, including A/D converters. The FEPCM provides an efficient means to significantly improve the communication performance of a VAX processor. The features and benefits of the FEPCM option are as follows:

- Provides the powerful performance and software of the PDP-11 for sophisticated communication applications
- Easily programmed for special-purpose applications
- Simplifies VAX processor communication tasks
- Provides multifunction capabilities

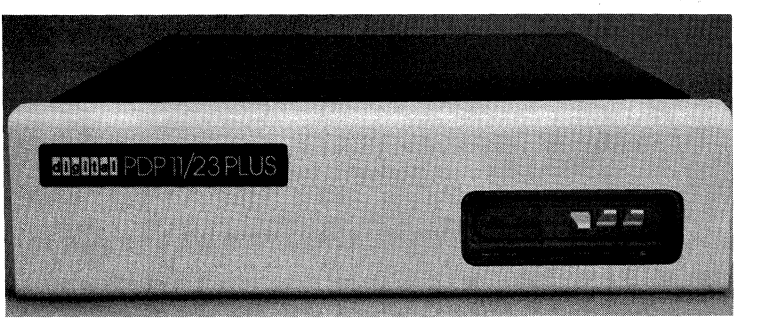


Figure 2-48 ▪ FEPCM PDP-11/23-PLUS Front-end Communications Processor

The FEPCM communication processor is available mounted in the H9642 cabinet or can be mounted in a standard cabinet or rack 48.26 centimeters (19.0 inches) wide. It contains the PDP-11/23-PLUS or PDP11-24 microproccessor and memory, UNIBUS backplane, and power supply. It connects to the VAX processor UNIBUS through a UNIBUS backplane connector and attached cable. Figure 2-49 shows the FEPCM and VAX processor configuration.

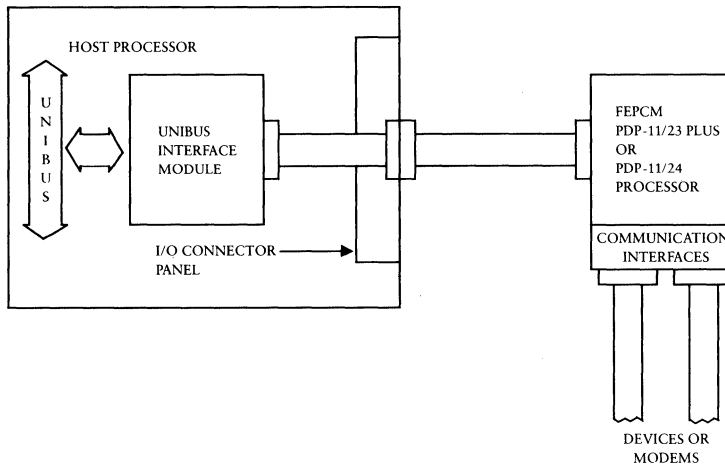


Figure 2-49 • FEPCM Communication Processor Configuration

• MODEMS AND MULTIPLE MODEM ENCLOSURES

Digital's modems are reliable communication devices that are used to transfer information between Digital's systems, terminals, and devices and many systems and terminals of other manufacturers through public or private leased telephone lines. The modems convert the digital data to analog signals for transmission through the lines and convert the analog signals to digital data at the remote location upon reception. The selection of a modem is determined by the transmission speed requirements, the type of data transmission and data characteristics, and the interface signals standards. Digital provides a complete series of modem units and modem enclosures for a variety of applications. The modem units are small tabletop devices that can be conveniently located and the modem enclosures enable many modems to be mounted in a single location. Figure 2-50 shows a typical local and remote system configuration using Digital's modems. Some of the modem features are:

- Small and reliable units fully supported by Digital's service organization
- Provide effective communications between many types of transmission devices
- Available for full- and half-duplex transmission
- FCC approved for direct connection to telephone lines

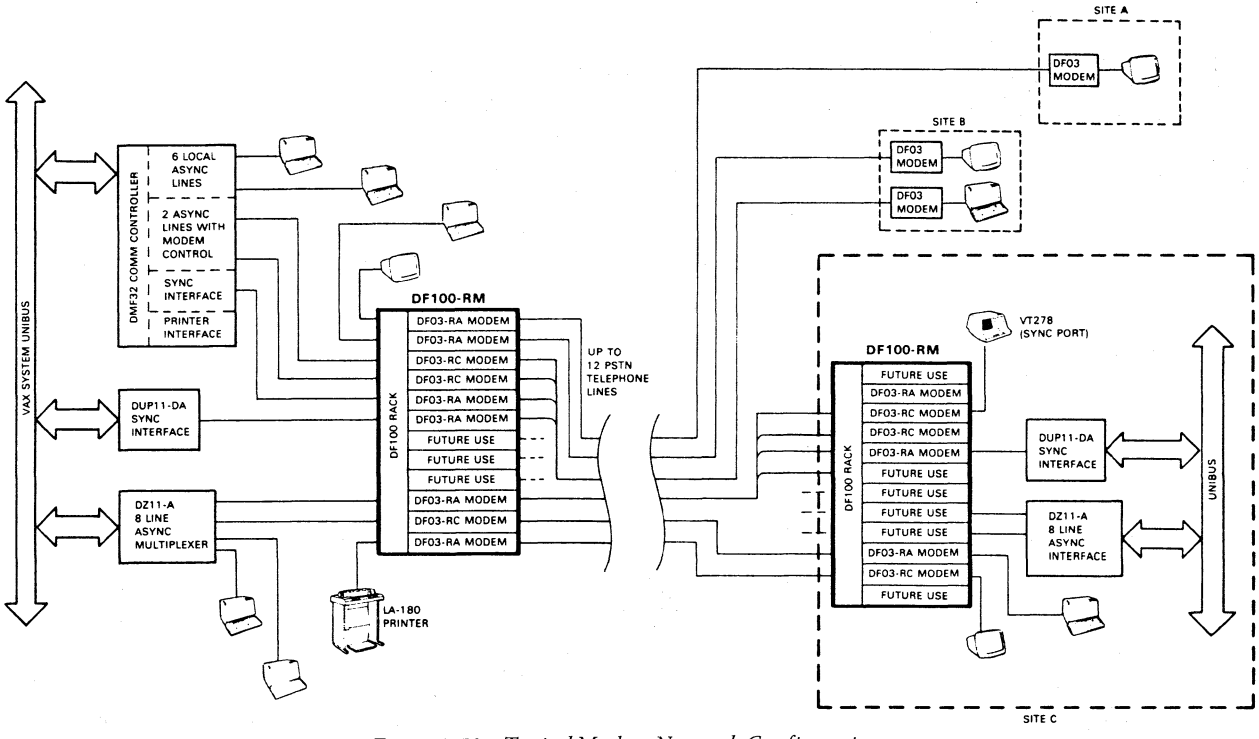


Figure 2-50 ▪ Typical Modem Network Configuration

DF02 and DF03 Modem Units—The DF02 and DF03 modems are small reliable units that can be conveniently positioned on a desk or table and provide full-duplex transmission through unconditioned dialup of two-wire leased lines. The units contain a power supply and connect directly to the telephone lines through a cable and RJ11C connector. An RJ11C receptacle located at the rear of the unit enables the unit to be connected to a standard telephone for both voice and data communication. Also mounted at the rear of the unit is a standard EIA connector, which permits the units to be connected to a computer or device that conforms to EIA standard RS-232-C. These connectors allow the units to be easily installed and removed. Both modems contain a row of push switches to select the transmission requirements and indicator lights that display the modem status. The modems are available with or without the automatic call unit (ACU) feature that permits calls to be initiated without dialing manually on a telephone.

DF02 modems permit full-duplex, asynchronous transmission at switch selectable data rates up to 300 bits per second. It includes local and remote test capabilities, and calls can be originated and answered manually or automatically under the control of the computer or device. It is compatible with all Digital asynchronous data communications controllers that support EIA RS-232-C standard interfaces. When used with a standard telephone, the DF02 offers alternate voice and data capabilities. The DF02 is physically similar to the DF03 modem and is compatible with the DF03 modem and with Bell 103J and 212J data sets at rates up to 300 bits per second. The DF02 modems are available in two configurations: the DF02-AA and the DF02-AC. The units contain the same features except that the DF02-AC contains the ACU and uses an asynchronous ASCII input format at data rates of 110, 300, and 1,200 bits per second and can store up to 16 digits for dialing and redialing.

The DF03 modem, shown in figure 2-51, allows full-duplex, asynchronous or synchronous transmissions at data rates up to 300 bits per second or at 1,200 bits per second. At the low speed rate, up to 300 bits per second, the data transferred is asynchronous, and at 1,200 bits per second the data can be either character asynchronous or bit synchronous. The DF03 modem is compatible with the Bell 103J data sets at low speeds, with the Bell 212A data sets at the 1,200 bits per second rate, and with all Digital communication controllers that support EIA RS-232-C standard interfaces and dialup modem control. The DF03 modem is available in two configurations: the DF03-AA and the DF03-AC. Both units contain the same features except that the DF03-AC includes the additional functions described for the DF02-AC modem.

Figure 2-51 ▪ DF03 Modem Unit

DF100 Multiple Modem Enclosure–The DF100 modem enclosure, shown in figure 2-52, can be positioned on a table or installed into a standard cabinet or rack 48.26 centimeters (19.0 inches) wide and occupies a vertical mounting space of 26.7 centimeters (10.5 inches). Up to 12 single-card modem modules can be installed into the enclosure and modems can be added to expand the communication facilities. The DF100 unit eliminates the cables normally required for individual modem units. The DF100 includes a power supply that provides power to the modem modules and includes space for a redundant power regulator that assumes the power load if the standard regulator becomes defective. The modem modules are installed from the front of the cabinet and the connectors for the device and telephone lines are located at the rear of the unit. The features of the DF100 multiple modem enclosure follow:

- Houses up to 12 modem modules and contains internal power supply and optional power regulator
- Allows modules to be easily installed and removed for service or modem expansion
- Simplifies cabling requirements
- Allows selection of modem module types

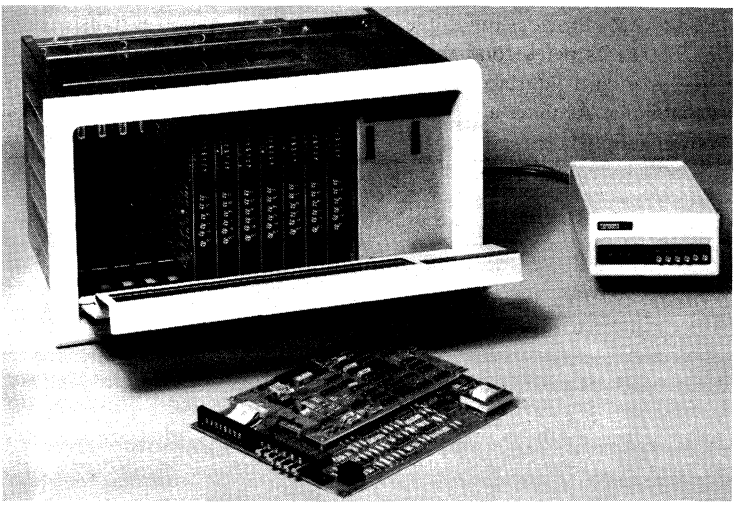


Figure 2-52 • DF100 Multiple Modem Enclosure

A complete series of modem modules are available for specific communication requirements. The following modules are available for either a public switched telephone network (PSTN) or private leased telephone network (PLTN).

- DF112-AM–Provides full-duplex, asynchronous or synchronous operation at transfer rates of up to 300 bits per second and at 1,200 bits per second. Contains integral auto-dialer and operates with dialup PSTN lines and two-wire leased line and multitype telephone service capabilities. Compatible with Bell 212A data sets and with CCITT V.22 standard interfaces.

- DF126-AM–Provides half-duplex, asynchronous, or synchronous operation at transfer rates of 2,400 bits per second and is compatible with Bell 201B data sets and with CCITT V.26 standard interfaces. Contains manual-originate/auto-answer or auto-originate/auto-answer capabilities. Also available for full-duplex operation through four-wire, leased-line applications.

- DF127-AM–Provides half- or full-duplex, synchronous operation at transfer rates of 4,800 bits per second through PLTN lines. Compatible with Bell 201B data sets and with CCITT V.26 standard interfaces. Contains auto-originate and auto-answer capabilities. Assumes a transfer rate of 2,400 bits per second if the integral power supply fails.

- DF129-AM – Provides full-duplex, synchronous operation at transfer rates of 9,600 bits per second through PLTN lines. Compatible with CCITT V.27 standard interfaces and provides auto-originate and auto-answer capabilities. Assumes a transfers rate of 2,400 bits per second if integral power supply fails.

Chapter 3 · VAX-11/725 and VAX-11/730 Processors

The VAX-11/725 and the VAX-11/730 central processing units (CPUs) are 32-bit, microprogrammed computers that execute the VAX instruction set in native mode and support the VMS operating system. The VAX-11/730 processor also supports ULTRIX-32 software, which is Digital's supported native UNIX™ operating system. Both processors execute nonprivileged PDP-11 processor instructions in the compatibility mode, allowing existing user-mode PDP-11 programs to run without modification.

The CPUs perform the logic and arithmetic operations requested by the computer system user. They use 32-bit addresses which provide access to 4.3 Gbytes of virtual address space. The processor's memory management hardware translates virtual addresses to physical addresses. The processor contains sixteen, 32-bit registers that can be used for temporary storage, as accumulators, index registers, and base registers. The native instruction set includes integer, packed decimal, character-string bit field, floating-point, and program control instructions and special instructions. The CPU can process the same data types as the larger-VAX processors, including bits, bytes, words, longwords, and quadwords. The floating-point accelerator option can be installed in either the VAX-11/725 or VAX-11/730 processor to decrease the instruction execution time of floating-point instructions and some integer arithmetic operations. The main features of the processors are:

™ UNIX is a trademark of AT&T Bell Laboratories.

- 32-bit microprogrammed VAX processor.
- Soft control store, which allows loading of microcode revisions and microdiagnostics.
- Programmed array logic (PAL), which increases logic density and reduces cost.
- PDP-11 compatibility mode.
- Optional floating-point accelerator.
- Dual TU58 cartridge tape drives as console loading devices.
- Small CPU packaging.

▪ VAX-11/725 Processor System

The VAX-11/725 is Digital's smallest and least expensive VAX system and includes the UNIBUS for I/O communications. It is a VAX-11/730 system specially packaged in a small cabinet 62.23 centimeters (24.5 inches) high, which includes an RC25 disk drive subsystem. The VAX-11/725 is designed for single-user workstation applications or economical multiuser systems. The unit is shown in figure 3-1 and includes the following features:

- Low heat dissipation and power requirements
- Small cabinet that can be positioned under a desk or table in an open office or similar environment
- Special power wiring and air conditioning not required
- Cabinet mounted on casters for easy relocating
- Hardware and documentation for customer installation
- Reliable mass-storage subsystem contained in CPU cabinet

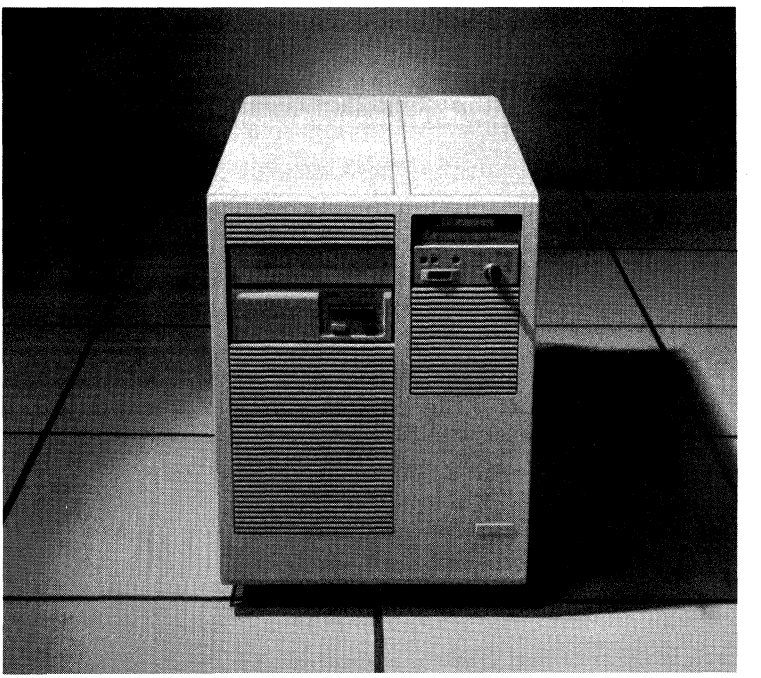


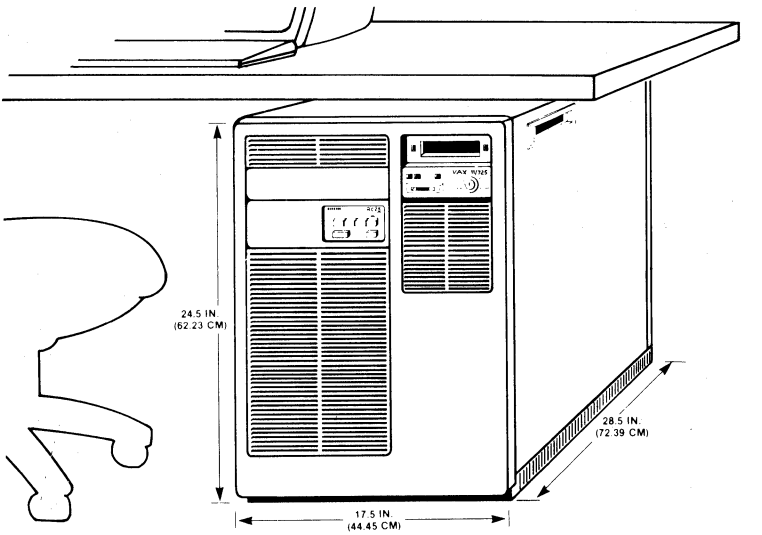
Figure 3-1 ▪ VAX-11/725 Processor Unit

The standard components included with the system are:

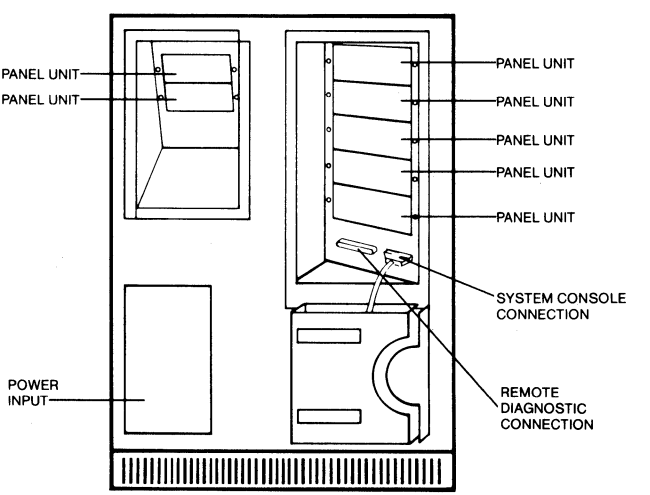
- VAX-11/730 CPU and power system
- 1 Mbyte of ECC MOS main memory
- Two TU58 tape cartridge drives
- RC25, 52-Mbyte, fixed and removable media disk subsystem
- Pedestal cabinet with rollers
- VAX/VMS operating system license

The CPU system backplane provides slots for UNIBUS option expansion; however, the UNIBUS cannot be expanded outside the system cabinet. One backplane slot is assigned to the floating-point accelerator (FPA) option module.

Figure 3-2 shows the VAX-11/725 cabinet dimensions and the I/O connector panel configuration at the rear of the unit. Seven spaces are available for mounting connector panel units, a console terminal connector, and a remote diagnostic connector. One of the panel units is dedicated to the cable connector for the VS100 graphics workstation terminal. The console terminal and remote diagnostic service modem are options.



Cabinet Front View



Cabinet Rear View

Figure 3-2 ▪ VAX-11/725 Processor Unit Configuration

RC25 Disk Subsystem

The RC25 disk subsystem is a reliable, 52-Mbyte, disk drive that uses Winchester technology and provides the flexibility of both fixed and removable disks. Twenty-six Mbytes of storage are contained on a nonremovable 8-inch disk and 26 Mbytes are stored on an 8-inch removable disk cartridge. A total of 50 Mbytes is available to the user. The RC25 subsystem includes an intelligent controller, microdiagnostics programs for automatic self-tests, and remote diagnostic testing. The subsystem provides most Digital Storage Architecture (DSA) capabilities and has the highest performance and capacity of any disk drive with fixed and removable disks.

VAX-11/730 Processor System

The VAX-11/730 is the smallest of the VAX systems that permit full UNIBUS device and mass storage expansion capabilities. It is designed for general purpose, realtime, and time sharing applications. The system is enclosed in a cabinet 53.9 centimeters (21.25 inches) wide, and 106.7 centimeters (42.0 inches) high. It contains two standard backplanes, one for the CPU, main memory and options modules and one assigned specifically to UNIBUS expansion. A VAX-11/730 packaged system is shown in figure 3-3. It features:

- A CPU and mass storage devices contained in a single-width, low-height system cabinet to reduce floor space and cable requirements.
- An integrated disk controller that operates with one R80 fixed-media and three RL02 removable-media disk drives.
- Memory expansion up to 3 Mbytes.
- A UDA50 disk controller for connecting up to four disk drives.

Figure 3-3 ▪ VAX-11/730 Processor Unit

A variety of CPU, mass storage, communications, and I/O options can be installed within the VAX-11/730 CPU cabinet or in a separate UNIBUS expansion cabinet. The standard system includes the following components:

- The CPU and power system
- 2 Mbytes of ECC MOS memory
- A mass storage device
- VAX/VMS software license

VAX-11/730 Disk Controllers and Drives

The RB730 integrated disk controller (IDC) mounts in a slot in the CPU backplane and controls the operation of an R80 disk drive and from one to three RL02 disk drives. It also can be connected to a total of four RL02 disk drives when the R80 disk drive is not included. Data is transferred between the IDC and CPU through the accelerator bus and is controlled in part by dedicated microcode in the CPU. During a read or write transaction, 32-bit longwords of disk data are transferred following the generation of a micro-level (fast) processor interrupt request by the IDC. Silo registers in the IDC provide up to 1 Kbyte of data buffering for both read and write data. The IDCs can also be connected to the UNIBUS for device interrupts other than the fast interrupts generated for disk data transfers. When connected to the UNIBUS, the IDC monitors the system's powerfail state.

The UDA50 is a microprocessor-based disk controller that operates with the UNIBUS and allows a total of four RA60, RA80, or RA81 disk drives to be connected in any combination. It provides Digital Storage Architecture (DSA) capabilities.

Figure 3-4 shows the cabinet dimensions and equipment configuration of VAX-11/730 packaged system.

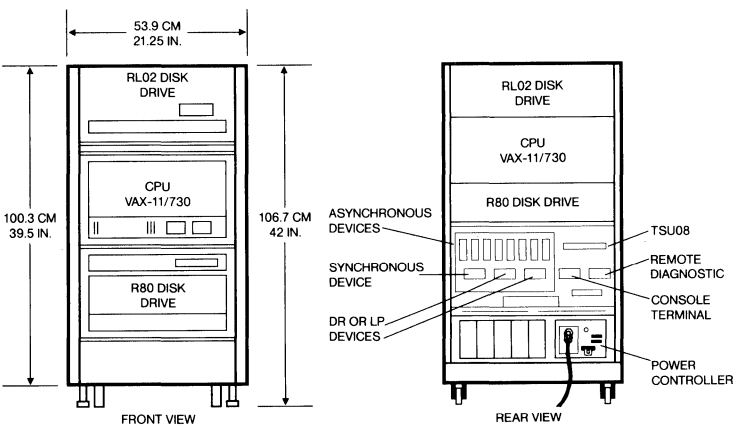


Figure 3-4 ▪ VAX-11/730 Packaged System Configuration

VAX-11/730 Processor Organization

The VAX-11/730 CPU logic consists of three hex-height modules: the data path (DAP) module, writable control store (WCS) module, and memory controller (MCT) module. The WCS module contains the console subsystem logic. A floating-point accelerator (FPA), contained on a hex-height module, is available and mounts in a dedicated slot of the CPU backplane. Programmed array logic (PAL) used in the CPU increases the logic density and reduces costs. It consists of high-density logic arrays contained on an integrated circuit (IC) and manufactured with TTL Schottky bipolar logic and fusable-link technology. The PALs are programmable AND circuits connected to fixed OR circuits. The circuit is configured by disconnecting fusable links within the IC. Figure 3-5 shows the organization of the VAX-11/725 and VAX-11/730 processor logic and internal bus structure.

The CPU microcontroller consists of a writable control store memory, a control store register, and a microsequencer. The writable control store is a programmable read or write memory that is loaded from the TU58 tape drive during the system bootstrap operation and provides storage for up to 20K microinstructions. The basic control store is a 16K by 24-bit dynamic RAM array and contains the microinstructions for the native-mode and PDP-11 compatibility-mode instructions. A 4K by 24-bit RAM array can be installed to support the integrated disk controller for user microcode. Each microinstruction is 24 bits long and contains several fields that control specific CPU functions. The microsequencer determines the sequence in which the microinstructions are read from the control store and loaded into the control store register for processing.

The CPU data path performs the arithmetic and logical operations necessary to execute the instruction set. The principal data path components are 4-bit processor slices. Eight of these slices are connected in parallel to give an arithmetic and logical processing element 32-bits wide. The data path also contains a 256 by 32-bit local storage RAM that includes the general registers and several of the architecturally defined privileged processor registers. The data path is controlled by the microcode that is executed in the CPU's microcontroller.

An address translation buffer contains frequently used virtual address translations. It significantly reduces the CPU time required for repetitive address translations. The buffer contains 128 virtual-to-physical page address translations consisting of 64 system space translations and 64 process-space translations. Each of these sections includes parity on each entry for increased data integrity.

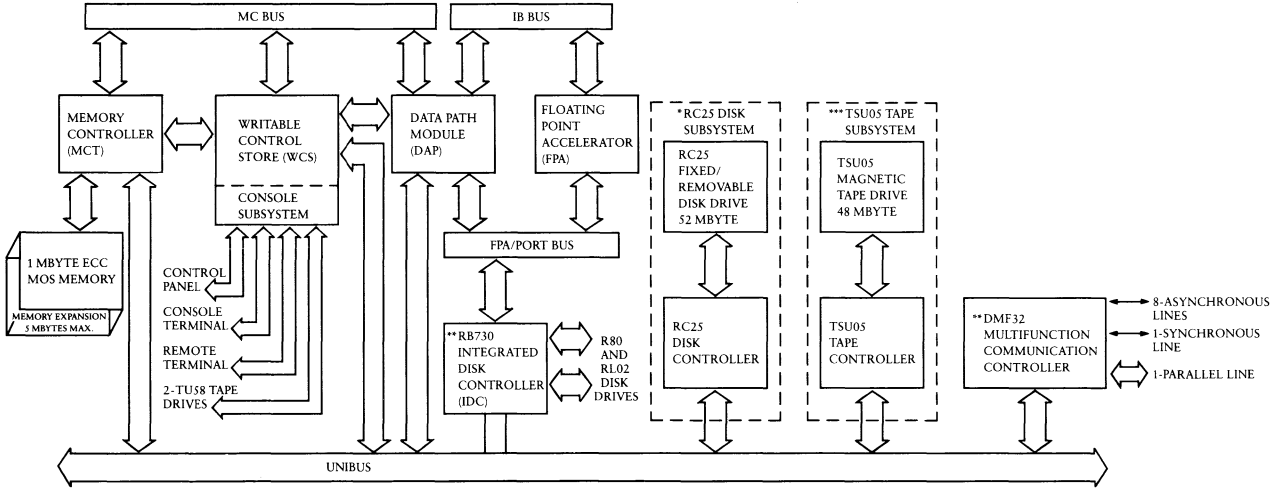


Figure 3-5 ▪ VAX-11/725 and VAX-11/730 Processor Organization

A one-longword prefetch instruction buffer allows the next instruction to be fetched while an instruction is executing. The control logic continuously fetches data from memory to keep the longword buffer full. In native mode, the variable length instructions are stored in contiguous byte positions in memory and are aligned on byte boundaries. In compatibility mode, the PDP-11 instructions are 16 bits long, occupy two contiguous bytes, and are aligned on word boundaries.

The VAX-11/730 processor contains an interval timer and a time-of-year clock. The interval timer is used to measure small time intervals and the time-of-year clock is used by software to perform various timekeeping functions.

▪ INTERNAL BUS STRUCTURE

Communications between the main logic elements in the CPU is through the internal bus structure, which consists of a memory control (MC) bus, memory array bus, console bus, FPA/port bus, and IB bus. External communication to the peripheral devices is through the UNIBUS.

The MC bus is bidirectional and connects to the WCS, DAP, and MCT module. It has 32 lines that transfer data and address information during memory and UNIBUS reference by the CPU. The data and address information is transferred during separate bus cycles. The MC bus is also used as part of the data path during nonprocessor request (NPR) transfers between a UNIBUS device and the memory array.

The memory array bus connects the MCT module to the memory array modules. One of five bus lines is enabled to select one of the five array modules, and fifteen bus lines are used to select the referenced memory location in the array. Two bank select lines are used to select a 64K location within the selected module.

The FPA/port bus connects the FPA module and the integrated disk controller (IDC) module to the CPU. The bus consists of 32 data lines, and the remaining lines provide data strobe and synchronizing signals to control the bus transfers.

An eight-line console bus is a buffered extension of the AD bus of the 8085A console microprocessor. It transfers 8 bits of data between the console processor on the WCS module and the data path logic on the DAP module.

The IB bus, which is part of the instruction processing logic on the DAP module, is an eight-line bus that connects to the FPA module. It allows the FPA to sample the 8-bit opcode data during the CPU's class decode operation.

- WRITABLE CONTROL STORE FUNCTIONS

The execution of the system-level instructions and operations is sequenced and controlled by the microprogram contained in the writable control store (WCS) module. The WCS also contains the console subsystem, which is the main operator and maintenance interface. The WCS consists of the following logic:

- Console subsystem
- CPU control store logic
- Clock generator and interval timer
- UNIBUS data transceivers

Console Subsystem–The console subsystem, shown in figure 3-6, consists of the front panel switches and indicators, the console terminal port, remote diagnostic port, two TU58 tape cartridge drives and interface, and the console-CPU interface logic. The 8085A microprocessor executes the console program that controls the console and terminal functions. A 4-Kbyte or 6-Kbyte ROM contains the resident portion of the program and a 16-Kbyte RAM stores the main part of the console program. The 6-Kbyte ROM is used when the remote diagnostic service option is included. The RAM, which is loaded from the TU58 drive during the system bootstrap operation, stores the microdiagnostic and microdiagnostic monitor program. The console terminal, remote diagnostic line, and TU58 tape drives each connect to the console subsystem through universal synchronous/asynchronous receivers and transmitters (USARTS).

Data and address information is multiplexed and transferred within the console subsystem through the AD bus. An 8-bit console write register and an 8-bit console read register, which are on the DAP module, are used to transfer the data between the console subsystem and the CPU.

A clock generator and interval timer provide the basic timing signals and timing intervals for the system operation. The interval timer is controlled by the microprocessor and provides the basic timing for console interrupts, power fail functions, and the time-of-year clock. The interval timer permits measurement of small time intervals. The time-of-year clock is used by software to perform various timekeeping functions.

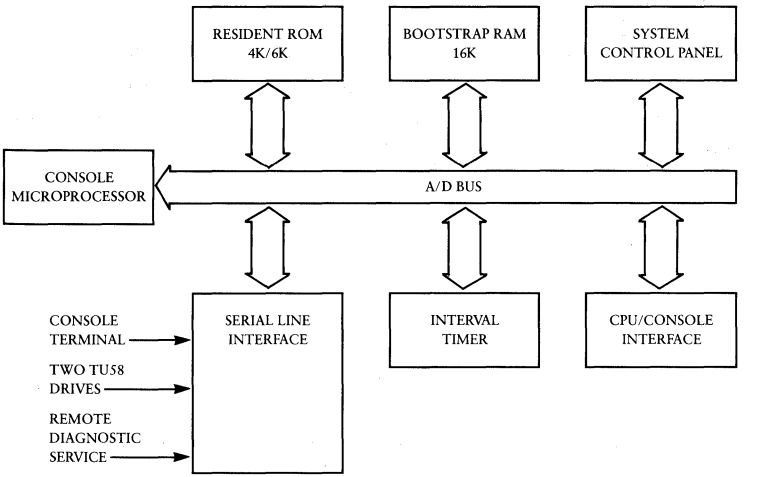


Figure 3-6 ▪ VAX-11/730 Console Subsystem

▪ DATA PATH FUNCTIONS

The data path logic on the DAP module performs the arithmetic and logical operations necessary to execute the instruction set. The logic consists of

- Data path processor slices
- Instruction and interrupt processing logic
- CPU microsequencer
- Console registers

The data path consists of eight 4-bit processor slices connected in parallel to provide a 32-bit processing element. The data path also contains a 256 by 32-bit RAM that includes general registers and several architecturally defined priviliged registers. The instruction and interrupt-processing logic is controlled by the microcode executed in the CPU microcontroller, which consists of the microsequencer and the soft control store on the WCS module. The one-longword prefetch instruction buffer is part of the instruction processing logic and allows the next instruction to be fetched while an instruction is executing. The control logic continuously fetches data from memory to keep the longword buffer full. In native mode, the variable length instructions are stored in contiguous byte positions in memory and are aligned on byte boundaries. In compatibility mode, the PDP-11 instructions are 16 bits long and occupy two contiguous bytes aligned on word boundaries.

The console registers are an 8-bit read register and an 8-bit write register and are used to transfer data, one bit at a time, between the console processor and the CPU data path.

- MEMORY CONTROLLER FUNCTIONS

The memory controller (MCT) module contains the memory and UNIBUS control logic. The memory control logic controls data transfers to and from the main memory array modules over the array bus. The UNIBUS control logic controls transfers to and from the peripheral devices over the UNIBUS. The memory control contains the following logic elements:

- Address translation buffer
- UNIBUS map and arbitrator
- Data rotator and substituter
- Microsequencer and control store
- Error correction logic (ECC)

Data transfers are initiated by the CPU data path under the control of the CPU microcode, or by the UNIBUS devices when direct data transfers are made between the devices and memory. The translation buffer converts virtual addresses to physical addresses. The physical addresses are then applied to the memory array modules. The address translation buffer contains frequently used virtual address translations. It significantly reduces the amount of time that the CPU requires for the repetitive task of dynamic address translation. The buffer contains 128 virtual-to-physical page address translations—64 system-space translations and 64 process-space translations. Each of these sections includes a parity check on each entry to ensure the data integrity.

The UNIBUS map converts 18-bit addresses to physical memory addresses. The UNIBUS arbitrator regulates activity on the UNIBUS and assigns the memory controller to the CPU if the controller is not being used by a UNIBUS device.

A 32-bit longword of data is always retrieved from the memory; however, the requested byte may not be in the proper position for transfer. The data rotating and substituting network aligns the memory data in the proper order.

The control store RAMs and associated logic supply 72-bit microwords that control operations within the memory controller. The memory microsequencer selects each subsequent 72-bit microword. The ECC logic detects and corrects single-bit errors when a longword is read from the memory array.

- MAIN MEMORY SUBSYSTEM

The main memory in the VAX-11/730 processor consists of from one to five 1-Mbyte memory array modules. A memory array contains four banks and each bank is composed of 39 64K by 1-bit metal oxide semiconductor RAM integrated circuits (ICs). Memory data is transferred over the array bus as a 32-bit longword (4 bytes) and seven associated ECC bits. Single-bit errors are corrected and double-bit errors are detected but not corrected. All errors are reported back to the CPU, where they may be recorded for use in preventive and corrective maintenance operations.

Basic Memory Operations–The physical address space contained in the memory controller is grouped into 15 Mbytes of physical memory addresses and 1 Mbyte of I/O device addresses. A physical address location is selected by 24 address bits to enable access to a total of 16 Mbyte locations. During CPU or I/O transfer operations, the system performs a read or a write operation to the addressed location. The hexadecimal address assignments of the physical address space are shown in figure 3-7.

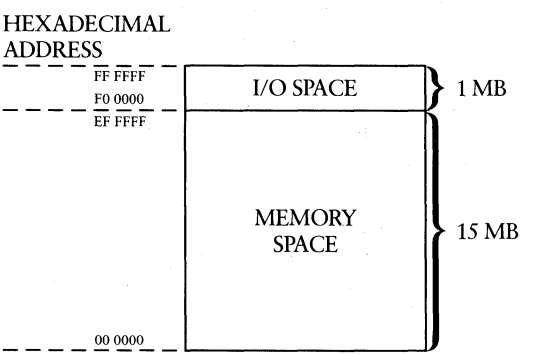


Figure 3-7 ▪ Memory Controller Physical Address Space

The error correction code (ECC) logic generates seven check bits from the 32-bit longword and appends these bits to the longword before it is stored in memory. Data read from memory is examined by the ECC logic to determine if an error has occurred. The ECC logic generates seven syndrome bits from the 39 retrieved bits to indicate no error, a single-bit error, or multiple-bit errors. The single-bit errors in the data are corrected using the seven syndrome bits and the CPU is informed that the data has been corrected. If a multiple-bit error is detected, the CPU is informed that the transferred data contains errors.

During a read operation, the physical address on the array bus selects a memory module and a location on the module. The addressed longword and its seven associated check bits from the specified location are placed onto the array bus. The ECC logic examines the longword and check bits for data errors. If a data error is found, the error is corrected and the corrected read data bit is set in the control and status register (CSR1). If the error cannot be corrected, the cycle is aborted and the read data substitute bit is set in CSR1. The ECC logic returns the longword to the array bus for the data rotator. The data is aligned, if necessary, and then sent to the MC bus.

When the CPU reads data from memory, the longword is sent to the CPU. When the CPU reads information from a UNIBUS device, the device data that was read is taken from the UNIBUS data lines, transferred to the MC bus through the writable control store transceivers, sent to the data rotator, and then transferred to the CPU. If a UNIBUS device initiates the read operation, a byte or word of data is transferred to the data lines of the UNIBUS.

When the CPU writes data, the data to be written is placed on the MC bus by the CPU and then sent to the data rotator. If the data is to be written to memory, the data rotator sends the data to the array bus. If the data is to be written to a UNIBUS device, the rotator sends the data back to the MC bus, which then places it on the data lines of the UNIBUS. When a UNIBUS device writes data to memory, the data is placed on the MC bus, sent to the data rotator, and then placed onto the data lines of the array bus. The ECC logic accesses the write data from the array bus and generates seven ECC check bits. The data and check bits are then returned to the array bus. The data and associated check bits are taken from the array bus and written in the selected array module.

▪ MEMORY CONTROL AND STATUS REGISTERS

Three control and status registers in the main memory subsystem provide control signals to the memory controller and report status information to the CPU. The control and status registers are CSR0, CSR1, and CSR2.

Memory Control/Status Register 0 – Memory control/status register 0 (CSR0) is a read-only register that contains the ECC check bits. The CPU reads the check bits to determine if a data error has occurred. Figure 3-8 shows the format of the CSR0.

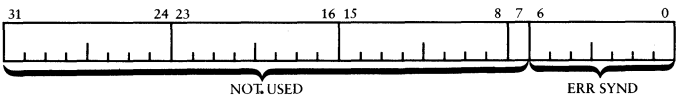


Figure 3-8 ▪ Memory Control/Status Register 0 Format

Bit	Function
31-7	Not used.
6-0	ERR SYND (Error syndrome) – These bits are syndromes if a correctable error occurred during a CPU-to-memory transaction, or are check bits if a noncorrectable error occurred.

Memory Control/Status Register 1 – Memory control/status register 1 (CSR1) is a read-write register. The CPU writes control bits into CSR1 for maintenance functions and to regulate the operations of the memory controller. Errors relating to a transfer between the CPU and memory are sensed by the error logic and error bits are set in the CSR1 to inform the CPU of the error condition. Figure 3-9 shows the format of the CSR1.

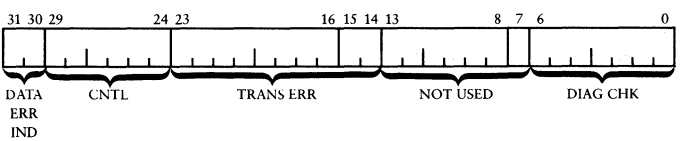


Figure 3-9 ▪ Memory Control/Status Register 1 Format

Bit	Function
31:30	DATA ERR IND (Data error indicators) – Set to indicate the following errors: Bit 31 Read data substitute – Set when an uncorrectable data error is detected during a CPU read-memory operation. Bit 30 Corrected read data – Set when a single-bit error has been detected and corrected during a CPU read-memory operation.
29:25	CTRL (Control) – Set to cause the following functions: Bit 29 Translation buffer parity diagnostic – When this bit is set, a parity error will occur during an access to the translation buffer. Bit 28 Inhibit reporting of corrected read data – Set to inhibit the reporting of corrected read-data errors. When this bit is set, the detected errors will be corrected by the ECC logic and syndromes will be logged in bits 6 and 0 of the CSR0; however, the corrected read data (bit 30) will not be set. Bit 27 Memory management enable – Set to enable address translations in the translation buffer. When cleared, the CPU address will be mapped directly into physical memory. The UNIBUS map is always enabled. Bit 26 Diagnostic check – Set to indicate that the memory controller is in the diagnostic mode. Bit 25 Disable ECC – Set to disable the ECC function of memory.
24	Not used.
23:14	TRANS ERR (Transactional error) – Set to indicate that an error has been detected during a read or write transaction as follows: Bit 23 Modify reference – Set to indicate that the modify bit was not set when an entry, referenced with a write-check-protection request, was made in the translation buffer . Bit 22 Access referenced – Set to indicate that the access requested by the CPU was not allowed when memory management is enabled. This bit is ignored if the translation buffer parity error (bit 15) is set. Bit 21 Translation buffer miss – Set to indicate that the translation buffer tag and bits 30:15 of the virtual address register (VAR) did not match or that the byte offset bit of the VAR was not set. Bit 20 Illegal UNIBUS operation – Set by an operation error to indicate that the CPU attempted to perform a memory operation in compatibility mode that was illegal or that the CPU attempted an illegal UNIBUS reference.

Bit	Function
	Bit 19 Write-across-page error – Set if a CPU write operation with a write check attempts to write across a page boundary. Bit 18 Adapter register select – Set to indicate that the physical address of a memory reference is located in the 768-Kbyte UNIBUS adapter space. Bit 17 UNIBUS busy – Set to indicate that a CPU has initiated a UNIBUS transaction and the UNIBUS is busy. Bit 16 Nonexistent memory – Set to indicate that the memory select decoder has referenced a memory address that does not exist. Bit 15 Translation buffer parity error – Set to indicate that a CPU access to the translation buffer resulted in a parity error. Bit 14 Translation buffer valid – Set to indicate that the TB entry valid bit is not set during a CPU access to the translation buffer.
13-7	Not used.
6-0	DIAG CHK (Diagnostic check bits) – Used to check the ECC logic.

Memory Control/Status Register 2 – Memory control/status register 2 (CSR2) is a read-only register. Errors relating to a transfer between the UNIBUS and memory are sensed by the error logic and the appropriate bits are set in CSR2. When the error logic causes a timeout on the UNIBUS, the CPU reads the error bits in CSR2. Figure 3-10 shows the format of CSR2.

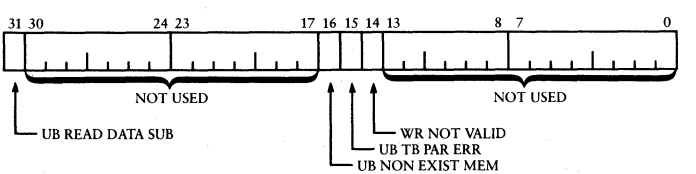


Figure 3-10 ▪ Memory Control/Status Register 2 Format

Bit	Function
31	UB READ DATA SUB (UNIBUS read data substitute) – Set to indicate that an uncorrectable error was detected during a UNIBUS read-to-memory transaction.
30:17	Not used.
16	UB NON EXT MEM (UNIBUS nonexistent memory) – Set to indicate that a UNIBUS request referenced a nonexistent memory location.
15	UB TB PAR ERR (UNIBUS translation buffer parity error) – Set to indicate that a TB parity error was detected during a UNIBUS to memory transaction.
14	WR NOT VALID (Write not valid) – Set during a two-cycle operation to indicate that an attempt to write into a memory page occurred and that the translation buffer did not contain a valid entry for the page.
13:0	Not used.

- PROCESSOR OPTIONS

The main memory of the VAX-11/725 and VAX-11/730 processor can be expanded by adding MA730 memory arrays modules. Each MA730 array module contains 1 Mbyte of ECC MOS memory.

The FP730 floating-point accelerator is a hardware option that operates with the VAX-11/725 and VAX-11/730 processors to execute the standard floating-point instruction set. Floating-point representation permits a greater range of number values than is possible with a 32-bit integer. The FP730 is a single, hex-height module that is easily installed in the CPU backplane. It is functionally transparent to the user. The use of the FP730 does not require hardware and software modifications; however, the CPU microcode must be reloaded from the TU58 tape drive.

The FP730 receives an opcode from the CPU and decodes the information into a starting microaddress. The control store RAM outputs control the arithmetic operations and data path logic. The FP730 option executes addition, subtraction, multiplication, and division instructions, which operate on single-precision (32-bit), double-precision (64-bit), extended-range double-precision (64-bits), and extended-range quadrupled-precision (128-bits) operands. It executes extended multiply (*EMOD*) and polynomial evaluation (*POLY*) instructions, and converts data between integer and floating-point formats and between single- and double-precision floating-point formats. The FP730 also executes integer multiplication and division instructions. The floating-point instructions performed by the FP730 are listed in table 3-1.

Table 3-1 ▪ FPA Floating, Double, and Integer Instructions

Opcode	Mnemonic	Opcode	Mnemonic
40	ADDF2	4BFD	CVTRGL
41	ADDF3	6AFD	CVTHL
60	ADDD2	6BFD	CVTRHL
61	ADDD3	56	CVTFD
40FD	ADDG2	99FD	CVTFG
41FD	ADDG3	98FD	CVTFH
60FD	ADDH2	76	CVTDF
61FD	ADDH3	32FD	CVTDH
42	SUBF2	33FD	CVTGF
43	SUBF3	56FD	CVTGH
62	SUBD2	F6FD	CVTHF
63	SUBD3	F7FD	CVTHD
42FD	SUBG2	76FD	CVTHG
43FD	SUBG3	4C	CVTBF
62FD	SUBH2	6C	CVTBD
63FD	SUBH3	4CFD	CVTBG
44	MULF2	6CFD	CVTBH
45	MULF3	4D	CVTWF
64	MULD2	6D	CVTWD
65	MULD3	4DFD	CVTWG
44FD	MULG2	6DFD	CVTWH
45FD	MULG3	4E	CVTLF
64FD	MULH2	6E	CVTLD
65FD	MULH3	4EFD	CVTLG
46	DIVF2	6EFD	CVTLH
47	DIVF3	51	CMPF
66	DIVD2	51FD	CMPG
67	DIVD3	71FD	CMPH
46FD	DIVG2	54	EMODF
47FD	DIVG3	54FD	EMODG
66FD	DIVH2	74FD	EMODH
67FD	DIVH3	55	POLYF
48	CVTFB	55FD	POLYG
68	CVTDB	75FD	POLYH
48FD	CVTGB	71	CMPD
68FD	CVTHB	74	EMODD
49	CVTFW	75	POLYD
69	CVTDW	C4	MULL2
49FD	CVTGW	C5	MULL3
69FD	CVTHW	C6	DIVL2

Opcode	Mnemonic	Opcode	Mnemonic
4A	CVTFL	C7	DIVL3
4B	CVTRFL	4F	ACBF
6A	CVTDL	6F	ACBD
6B	CVTRDL	4FFD	ACBG
4AFD	CVTGL	6FFD	ACBH

Chapter 4 · VAX-11/750 Processor

The VAX-11/750 processor, shown in figure 4-1, is a midrange member of the VAX family. It implements the VAX architecture and operates with VAX/VMS operating system and associated layered software or with ULTRIX-32, a Digital enhanced UNIX-based operating system. The VAX-11/750 is an efficient processing system that is designed for the concurrent operation of many users in performing timesharing or batch processing, or for realtime applications. Some of the features of the VAX-11/750 systems follow:

- Provides full VAX power in a relatively small cabinet
- Operates with UNIBUS and MASSBUS devices
- Can include up to 8 Mbytes of main memory
- Incorporates custom gate array technology to reduce the physical size and power consumption of the CPU and increase system reliability
- Provides 4 Kbytes of direct mapped memory to improve memory cache access time and increase overall performance
- Offers a user-control-store option to increase throughput by allowing the user to create routines in microcode tailored to specific applications
- Offers a floating-point accelerator option to decrease time required for floating-point arithmetic and for some integer-arithmetic operations

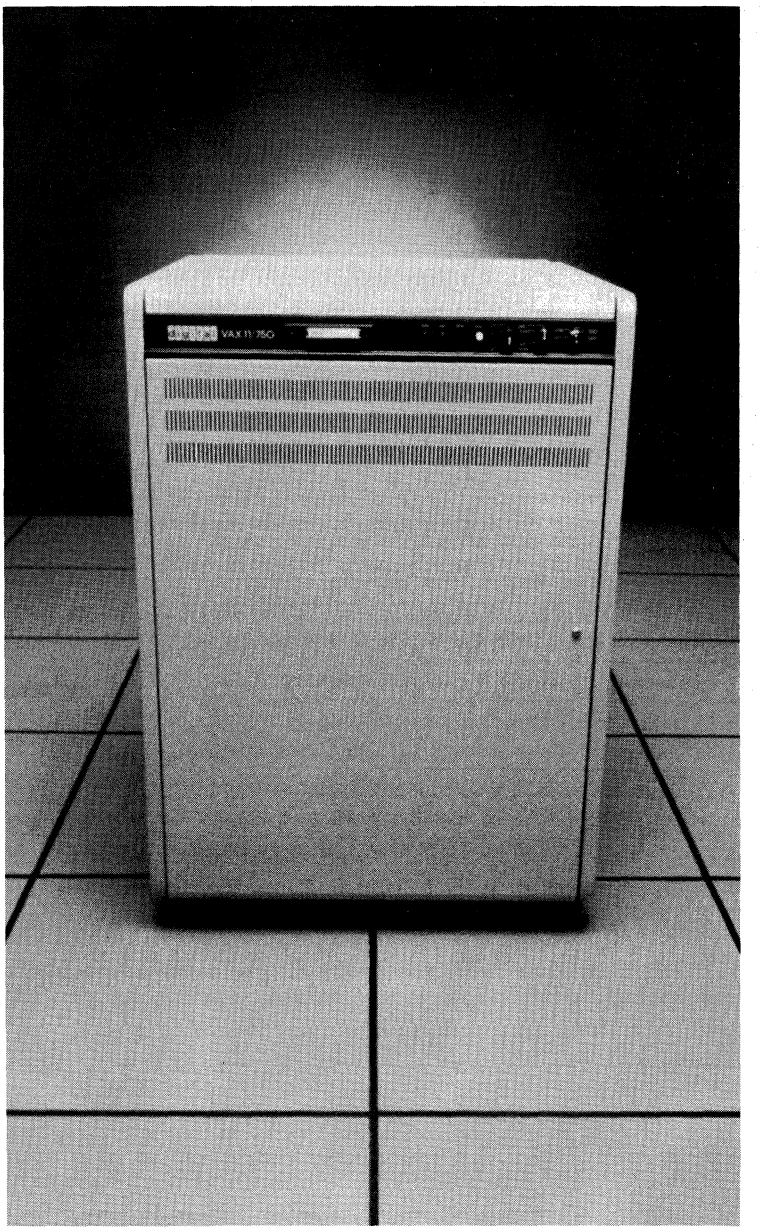


Figure 4-1 ▪ VAX-11/750 Processor Unit

• VAX-11/750 Processor System

The VAX-11/750 processor is a 32-bit, synchronous microprogrammed computer that executes variable-length instructions in native mode and nonprivileged PDP-11 instructions in compatibility mode. Execution of logic and arithmetic operations is controlled by the program. The use of 32-bit virtual addresses allows access to over 4 gigabytes of virtual address space. The processor's memory management hardware translates virtual addresses into physical addresses. The processor includes sixteen 32-bit registers that can be used for temporary storage and as accumulators, index registers, and base registers.

The native instruction set is highly versatile and bit efficient. It includes integer, packed-decimal, character string, bit field, and floating-point instructions; and program control and special instructions. The instructions and data can vary in length and can start on any byte boundary or, in the case of bit fields, at any arbitrary bit in memory.

The CPU processes data in the form of bits, bytes, words, longwords, and quadwords. It also processes 32-bit, single-precision and 64-bit, double-precision floating-point data; packed-decimal data up to 31 digits; and character-string data of up to 64 Kbytes. The standard components included with the processor are

- CPU control store
- Internal data paths
- 4-Kbyte direct-mapped memory cache
- 512-entry address translation buffer
- 8-byte prefetch instruction buffer
- Time-of-year clock and programmable realtime clock
- Optional floating-point accelerator
- Optional 10K by 80-bit user-control-store memory

Most of the logic circuits in the VAX-11/750 CPU consist of custom LSI (large-scale integration) gate arrays. Gate arrays are high-density logic circuits that use low-power Schottky technology. The circuits are configured by Digital during manufacture to conform to the circuit requirements. Compared with conventional circuits, gate arrays increase the speed and decrease the physical size and power consumption of the logic circuits. Because fewer components are necessary to implement the same functions, the system reliability is significantly increased.

VAX-11/750 System Configuration

The VAX-11/750 processor is contained in cabinet 106.7 centimeters (42.0 inches) high and 76.3 centimeters (29.0 inches) wide. The cabinet, shown in figure 4-2, is on casters and can be easily moved. The control panel and TU58 cartridge tape drive are on the front of the cabinet for easy access. The cabinet contains a CPU and memory backplane and a UNIBUS backplane. The CPU and memory backplane provides space for installing up to eight 1-Mbyte memory modules, a remote diagnostic module, a floating-point accelerator module, and a writable-control-store module. Three spaces are available for I/O expansion and can include UNIBUS adapters and MASSBUS adapters. The UNIBUS expansion backplane has seven slots in which I/O device adapters and interface modules can be installed. The CPU cabinet also includes a space for the memory battery backup unit and an I/O connection panel area at the rear of the unit for mounting I/O device connectors.

Two H9642 UNIBUS expansion cabinets, shown in figure 4-3, are available to extend the UNIBUS from the system cabinet. Both UNIBUS cabinets are 106.7 centimeters (42.0 inches) high and 53.9 centimeters (21.25 inches) wide. An RL02, R80, or R81 disk drive and BA11-K mounting box can be mounted in the H9642-FA (120-volt) and H9642-FB (240-volt) expansion cabinets and a BA11-K mounting box can be installed in the H9642-FC (120-volt) and H9642-FD (240-volt) expansion cabinets. The DD11-CK and DD11-DK UNIBUS backplanes can be installed in the BA11-K mounting box.

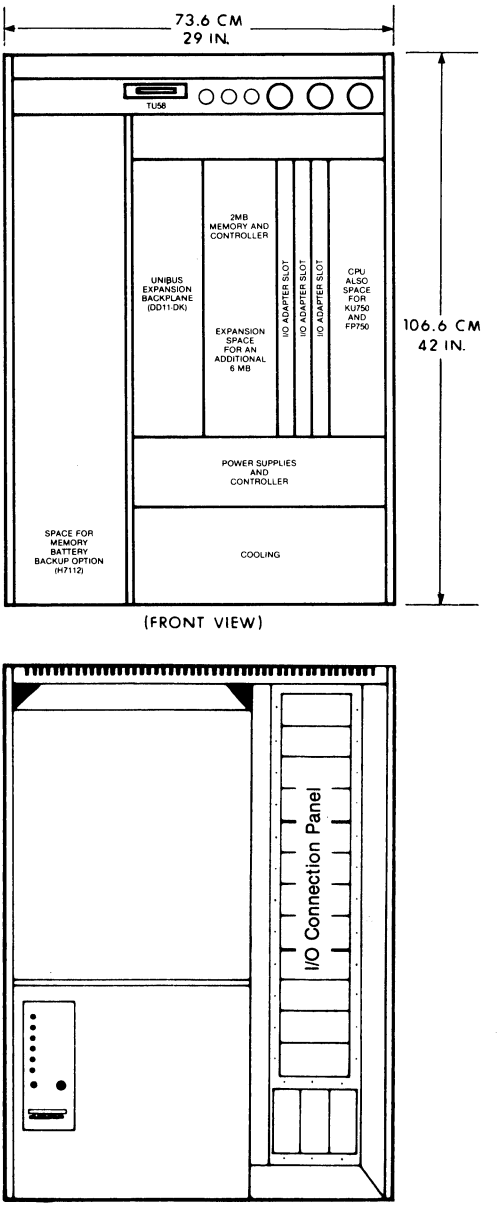


Figure 4-2 • VAX-11/750 Processor Cabinet Configuration

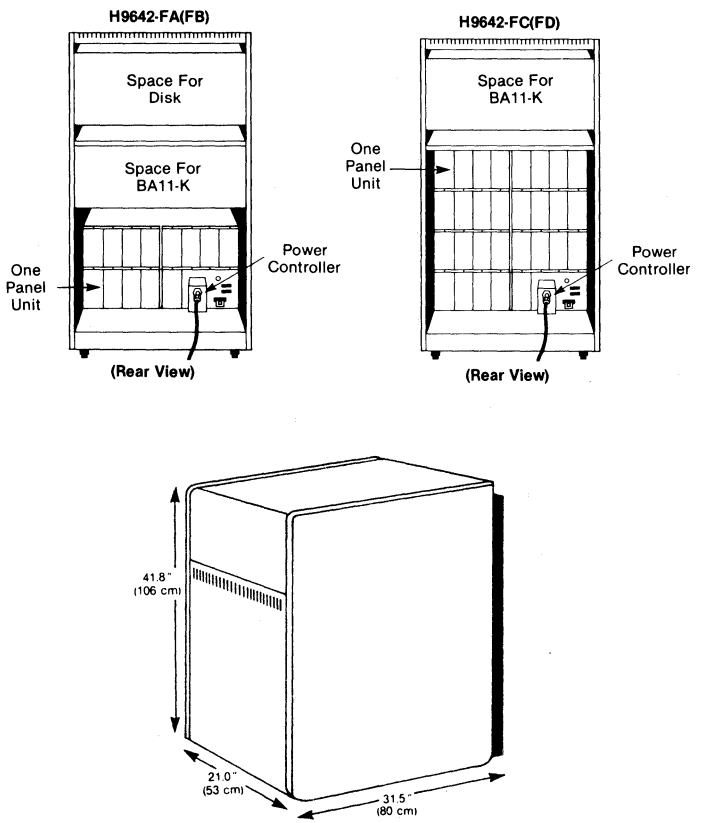


Figure 4-3 ▪ H9642 UNIBUS Expansion Cabinet Configuration

VAX-11/750 Processor Organization

The VAX-11/750 CPU logic is contained on the data path (DAP) module, the UNIBUS interconnect module, the memory interconnect (MIC) module, the memory controller (MCT) module, and the CPU control store module, shown in figure 4-4. The UNIBUS interconnect module includes the console subsystem and asynchronous serial-line interfaces for the TU58 cartridge tape drive and console terminal. The remote diagnostic module (RDM) and the floating-point accelerator (FPA) module are optional.

The CPU operations are controlled by a programmable read-only memory (PROM), which contains the native-mode and PDP-11 compatibility-mode instruction sets. A high-speed cache memory stores a copy of frequently selected information from the main memory for fast access to instructions and data. The processor includes an address translation buffer that contains frequently used address translations to reduce the time required for dynamic address translation. An 8-byte prefetch instruction buffer improves the CPU performance by accessing longword data in the instruction stream and storing the instructions prior to their actual use.

The VAX-11/750 also contains a programmable realtime clock and a time-of-year and data clock. The time interval of the realtime clock permits the measurement of finely resolved variable intervals. The time-of-year clock is used by the software to perform timekeeping functions and provide the correct time to the system in the event of a power failure or halt condition.

- INTERNAL BUS STRUCTURE

Communication between the main logic elements of the processor is through the internal bus structure, which consists of the W bus, M bus, memory-array bus and CMI bus. The CMI bus transfers information internally and externally to the I/O interfaces.

Under control of the microcode, information is transferred through 32 data lines of the W bus. The W bus connects to the UNIBUS interconnect logic and to the DAP logic, MIC logic, FPA logic, and remote diagnostic logic.

The M bus, consisting of 32 data lines, transfers information among the DAP, MIC, and FPA logic under control of the microcode.

The CMI bus provides 45 bidirectional lines for transfers of address, data, and priority arbitration information among all the logical elements of the CPU.

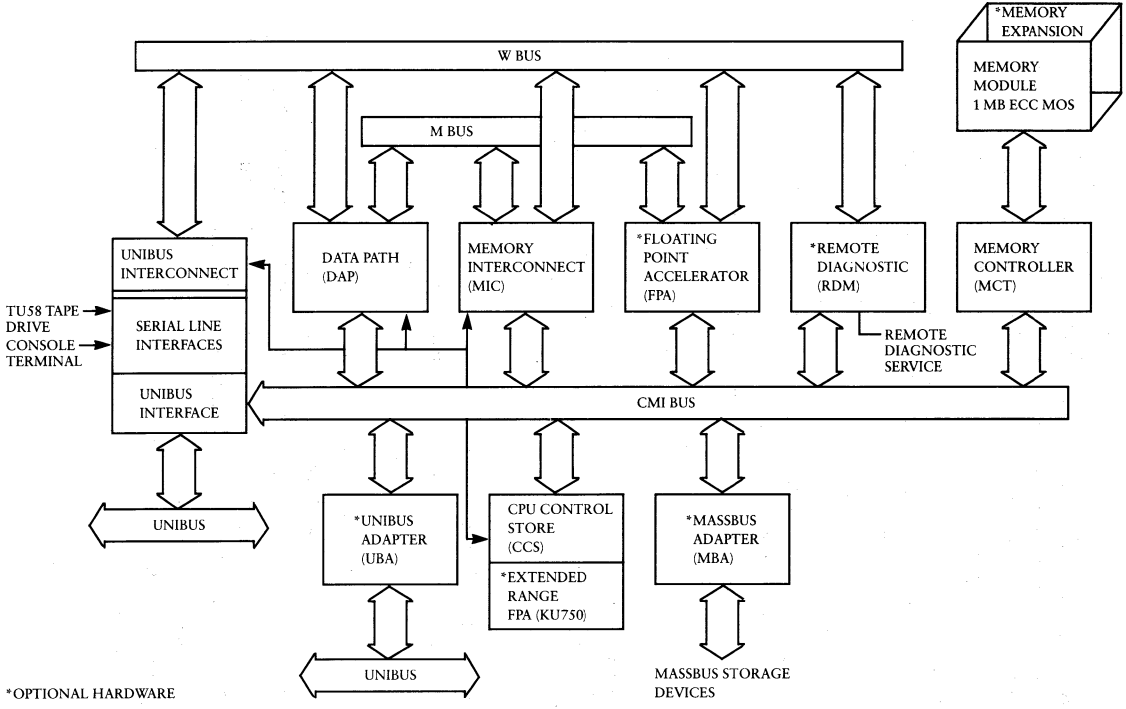


Figure 4-4 ▪ VAX-11/750 Processor Configuration

- PROCESSOR LOGIC ELEMENTS

The VAX-11/750 processor consists of the following logic elements contained on separate modules:

- Data path module – An extended-length, hex-height module that includes arithmetic logic, rotator logic, microsequencer logic, scratchpad registers, and interval timer.
- UNIBUS interface module – An extended-length, hex-height module that contains the TU58 tape drive and console terminal interface, interrupt logic, time-of-year clock, and UNIBUS interface.
- Memory controller module – An extended-length, hex-height module that contains the address logic, translation buffer, execution buffer, cache memory, and data routing and alignment circuits.
- CPU control store – An extended-length, hex-height module that contains the control store PROMs. The writable control store (KU750 option) mounts on this module.

Data Path Functions – The data path logic includes a microsequencer that provides an address to the control store PROMs for selecting the microinstruction to be performed. It performs the arithmetic and logic functions of the CPU. In addition to the microsequencer, the data path contains scratchpad memory, rotator logic, and an arithmetic logical (ALU) processor.

The scratchpad memory is composed of sixty-four 32-bit locations that are used for the general purpose registers (GPRs), the internal processor registers, and temporary storage registers. The scratchpad memory includes address control logic to enable access to the register information. Data is written into the scratchpad locations from the W bus.

The rotator logic receives a 64-bit input from and provides a 32-bit output to an internal DAP bus. The rotator performs field extractions, rotates and shifts data on the M bus and internal bus, and packs and unpacks floating-point data.

The ALU performs arithmetic or logical operations on binary and binary-code-decimal (BCD) 32-bit data.

CPU Control Store Functions – The CPU control store microcode, contained in ROMs, is used to sequence the CPU operations. The information is stored in six banks, each containing 1K 80-bit locations. One of the six banks is enabled by the bank select decoder, which receives information from the DAP logic. The control store RAMs receive information from the CMI bus and transfer control signals to the DAP, MIC, and UBI modules.

UNIBUS Interconnect Functions—The UNIBUS interconnect (UBI) logic contains the interfaces for the TU58 tape drive and console terminal, the interrupt control logic, a time-of-year clock, and a UNIBUS interface. Information is transferred to and from the UNIBUS interconnect through the W bus and CMI bus.

The interface for the console terminal and TU58 tape drive each contain an asynchronous EIA serial-line that operates under control of the microcode to transfer information between the console registers and the CPU and memory. The primary path for the data exchange is through the W bus.

The interrupt logic controls all hardware and software interrupts generated by the system. The interrupt logic stores sections of the processor status longword (PSL), including the interrupt priority level, the interrupt stack flag, and the current mode information. It also contains the asynchronous system trap (AST) level and returns this information to the system through the W bus under control of the microcode. The logic controls the UNIBUS arbitration by encoding the highest priority interrupt that is pending.

The functional elements of the UNIBUS interface include the UNIBUS data path and control, the address data path, the control store, and the UNIBUS arbitrator. The interface controls the UNIBUS and CMI bus protocols and monitors and controls the data transmissions between these buses. The control store consists of a 256 by 24-bit PROM that executes and directs the interface operations. A 512 by 19-bit address map RAM is loaded by the software and allows UNIBUS devices to access noncontiguous pages in main memory during DMA transfers. The timing and synchronization functions of the interface are provided by the CPU clock signals.

The time-of-year clock is a crystal-controlled, 32-bit binary counter that provides the correct time to the system for scheduling system operations. The correct time is initially entered by the operations system service and no operator intervention is required thereafter. The time is monitored by the software during timekeeping functions. The clock is equipped with a battery to insure that the correct time is maintained if the power fails.

Memory Interconnect and Control Functions—The memory interconnect and control (MIC) logic is the interface between the W bus, M bus, and the CPU interconnect (CMI) bus. The MIC module contains address control logic, a translation buffer, and cache memory, and performs the routing and alignment of memory data.

The address control logic contains the program counter and virtual address registers that store addresses for operand and instruction stream references and logic for address manipulations.

The translation buffer is a cache memory that stores 512 frequently used virtual-to-physical address translations, thereby reducing the time required by the CPU to perform these translations. The translation buffer is divided into 256 system-space page translations and 256 process-space page translations. It is two-way associative and parity is assigned on each entry to increase data integrity. The page table entries for virtual-to-physical address translations are stored by memory management microroutines. The translation buffer information is contained in an address field and a data field. The address field contains virtual address translations and the data field contains the physical page frame number for each page table entry.

Cache memory is a high-speed memory that maintains a copy of frequently selected portions of main memory for faster access to instructions and data. The cache memory is loaded by memory management and the processor examines the data in cache before accessing main memory. If the data is found in cache, the time for access is reduced. The cache is a 4-Kbyte, direct mapped, write-through memory that receives all memory information, including addresses and instructions. The cache data is longword-aligned with the CMI bus data, and the rotation for the data path alignment is performed by the memory data routing and alignment logic. The write-through feature protects the integrity of the memory information by updating the memory contents immediately after the CPU performs a write operation. For increased integrity, the cache includes byte parity for both data and addresses.

Main Memory Subsystem

The main memory subsystem consists of from one to eight 1-Mbyte memory array modules, the memory array bus, and the memory controller module. The memory modules contain dynamic 64-Kbit array MOS memory devices connected to the system through an error-correcting controller. The memory subsystem communicates with the CPU, UNIBUS, and MASSBUS through the internal CMI bus. The features of memory subsystem are as follows:

- Error correcting enhances data availability and reliability by correcting single-bit errors and detecting double-bit errors within the memory system.
- The subsystem permits write operations to any combination of bytes within an aligned longword.
- Boot ROMs allow bootstrapping from devices, including the integral TU58 cartridge tape drive.
- Optional battery backup maintains power to preserve data in 8 Mbytes of memory for a minimum 10 minutes.

▪ MEMORY CONTROLLER FUNCTIONS

The memory controller (MCT) contains the logic that is the interface between the main memory and the system. The memory controller module contains addressing logic, refresh control circuits, and error correcting code (ECC) logic. The memory subsystem includes diagnostic features that are controlled by the software.

The ECC logic corrects single-bit memory errors, detects double-bit errors, and detects errors greater than double-bit errors if there is an even number of errors. If a single-bit error is detected, the ECC logic corrects the error. The ECC logic detects multibit errors but no correction is performed. An uncorrectable error is a word containing an error that maps to an invalid syndrome. The address and syndrome of an uncorrectable error will overwrite the address and syndrome of a correctable error. Detections and corrections of errors are reported to the CPU, where they can be recorded for analysis.

The controller contains three longword registers that are accessible in the I/O space. The program monitors these registers to determine the memory size, record the address of failing memory locations, isolate the error to an individual memory circuit, and verify the operation of error-correcting logic. Memory performance is optimized through longword read and write operations. The memory controller also is designed to allow write operations to any combination of bytes within an aligned longword.

The memory controller can include four 256-Kbyte ROMs that contain the bootstrapping programs for up to four devices. The CPU reads the data from the ROMs into main memory and executes the program during the bootstrap loading operations. Each ROM contains a checksum of the ROM's contents to allow verification of proper operation and associated logic. One ROM always contains the bootstrap program for the TU58 tape cartridge and the remaining ROMs contain programs for other various system devices used to boot the system. The ROMs are selected by the boot switch on the control panel.

▪ BASIC MEMORY OPERATIONS

The physical address space contained in the memory controller is divided into memory addresses and I/O addresses. Figure 4-5 shows the combined areas. The three types of memory cycles used to address this space are read, longword-write, and byte and word write.

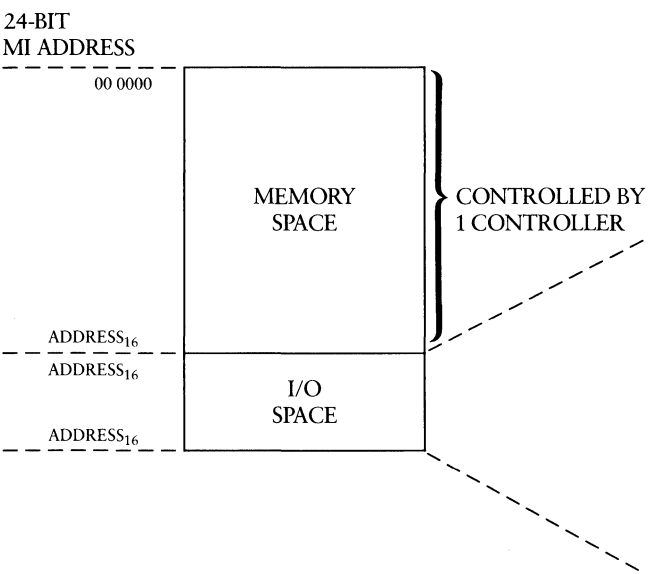


Figure 4-5 • Memory Controller Physical Address Space

During a *read cycle*, the device requesting the data transfers address and control information to the memory controller. The memory controller initiates the memory-cycle timing, transfers the memory data to the ECC logic, and notifies the device that information is available. If a correctable error is detected by the ECC logic, the corrected data is sent to the device and an interrupt is generated to notify the system that an error has occurred. If an uncorrectable error is detected, the unmodified data is sent to the device together with an error message. When the read reference is from an I/O adapter, the adapter records the error message. When it is from the CPU, the CPU will generate a machine check abort.

During a *longword write cycle*, the device requesting the longword write operation transfers address and control information to the memory controller. The memory controller initiates the memory-cycle timing, generates seven check bits, and transfers the bits and the 32-bit longword to memory.

During a *byte-* and *word-write cycle*, the device requesting the write operation transfers the address and control information to the memory controller and the controller initiates the memory-cycle timing. The memory controller forms the new 32-bit data word and transfers the longword and check bits into memory. If a correctable error is detected by the ECC logic, the error will be corrected. If the error is in a byte or word that is being written, the error will be ignored. When an uncorrectable error is detected by the ECC during the read portion of the cycle, the original longword will be rewritten into memory without any changes. The error will be reported during a subsequent read operation to this location.

- MEMORY CONTROL AND STATUS REGISTERS

Three control and status registers provide information for recording correctable and uncorrectable errors, for aiding the verification and diagnostics of the error checking hardware, and for indicating memory size and configuration. The registers are CSR 0, CSR 1, and CSR 2.

Memory Control/Status Register 0– Memory control/status register 0 (CSR 0) is a read-only register that contains information to allow the recording of both correctable and uncorrectable errors. The CSR 0 register format is shown in figure 4-6.

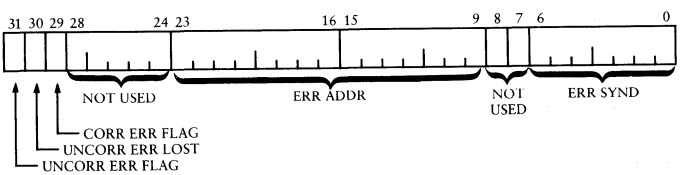


Figure 4-6 ▪ Memory Control/Status Register 0 Format

Bit	Function
31	UNCORR ERR FLAG (Uncorrectable error flag) – Set to indicate that an uncorrectable error has been detected and can be cleared only when UNCORR ERR LOST (bit 30) is cleared or if both bits are cleared during the same write operation.
30	UNCORR ERR LOST (Uncorrectable error lost) – Set to indicate that an uncorrectable error has occurred with the UNCORR ERR FLAG bit 31 previously set. The second uncorrectable error-page address will not overwrite the first uncorrectable error-page address. It is cleared by writing a one to this bit.
29	CORR ERR FLAG (Correctable error flag) – Set when a correctable single-bit error occurs during a read operation. A single-bit error that occurs during the read portion of a byte-write cycle will have no effect on the flag. Cleared by writing a one to this bit.
28:24	Not used.
23:9	ERR ADDR (Error address) – A hexadecimal address that identifies the page on which an error occurred during a memory read cycle. The page address corresponds to the first error that occurs, except that the page address of an uncorrectable error will overwrite the page address of a correctable error. The page address of an uncorrectable error is not overwritten by a more recent uncorrectable error.
8:7	Not used.
6:0	ERR SYND (Error syndrome) – Read-only bits that indicate the error syndrome of the detected error.

Memory Control/Status Register 1 – Memory control/address register 1 (CSR 1) is a read-write register that aids in the verification and diagnosis of the error correcting hardware. The format of CSR 1 following a powerup sequence is shown in figure 4-7.

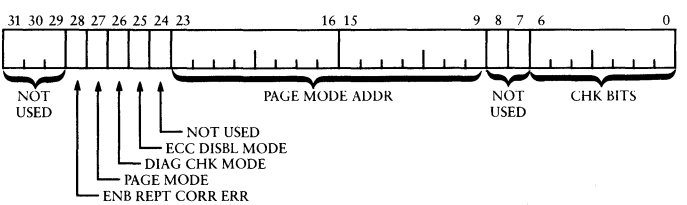


Figure 4-7 • Memory Control/Status Register Format

Bit	Function
31:29	Not used.
28	ENB REPT CORR ERR (Enable reporting correctable errors) – Set to enable the reporting of correctable single-bit errors and cleared to inhibit the reporting of single-bit errors. The errors will be corrected in either state.
27	PAGE MODE (Page mode) – Set to allow the ECC disable mode or the diagnostic-check mode to operate on a 512-byte page whose address is contained in CSR 1 (bits 23:9). The diagnostic-check mode operates in the page mode. The ECC disable mode can operate on a page or on the entire memory, depending on whether page mode is selected.
26	DIAG CHK MODE (Diagnostic check mode) – When set, the check bits 6:0 are substituted for the check bits that come from memory during a read operation. The correct check bits are transferred to memory during a write cycle. The diagnostic check mode is confined to a single page whose address is stored in the page mode address, bits 23:9. While operating in the diagnostic check mode, read errors that occur in other pages of memory will not be logged into CSR 0. The ECC disable mode and diagnostic check mode cannot be selected at the same time.
25	ECC DSBL MODE (ECC disable mode) – Set to disable the following: the detection of uncorrectable errors, the detection and correction of single-bit errors, the error logging in CSR 0, and the reporting of errors on the status lines. During a read operation, the generated check bits that are written into memory are also stored in CSR 0. The ECC can be disabled for a 512-byte page or for the entire memory, depending on whether the page-mode bit 27 is selected. The ECC disable mode and diagnostic-check mode cannot be selected at the same time.
24	Not used.
23:9	PAGE MODE ADDR (Page mode address) – In the page mode, this field contains the address of the 512-byte page to which the diagnostic check mode or the ECC disable mode will be applicable.
8:7	Not used.
6:00	CHK BITS (Check bits) – In the diagnostic check mode, these bits are substituted for the check bits from the memory during a read operation. In ECC disable mode, these seven check bits are read from memory.

Memory Control/Status Register 2 – Memory control/status register 2 (CSR 2) is a read-only register that provides memory size and configuration information. The format of the register information is shown in figure 4-8.

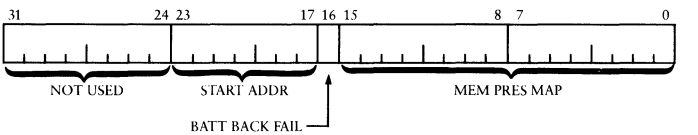


Figure 4-8 ▪ Memory Control/Status Register 2 Format

Bit	Function
31:24	Not used.
23:17	START ADDR (Starting address) – The starting address for the memory selected by jumper leads on the module.
16	BATT BACK FAIL (Battery backup failure) – Set on a powerup sequence if the memory battery backup unit voltage is not within the required value. Cleared by a longword write to any location in memory.
15:0	MEM PRES MAP (Memory present map) – These bits indicate the amount of memory available and their locations in the memory in the backplane. The map is divided into eight pairs of two bits.

▪ PROCESSOR OPTIONS

Several processor options can be added to improve the processor performance and extend the I/O capabilities.

The MS750 memory modules contain 1 Mbyte of ECC MOS memory and can be installed to increase the main memory to a total of 8 Mbytes.

The H7112 memory battery backup unit can be installed in the CPU cabinet to protect the contents of the MOS memory during power failures for a minimum of 10 minutes.

The KU750 writable control store (WCS) is an extended-range G and H data type option supported in KU750 loadable microcode. The module contains 1 K by 80-bit storage and attaches to the resident control store (CCS) module. It enables microroutines to be generated for user applications.

The remote diagnostic option is an extended-length, hex-height module that installs in the CPU backplane and allows diagnostic programs to be performed under control of Digital's Remote Diagnostic Service centers.

The FP750 floating-point accelerator option operates in conjunction with the VAX-11/750 processor to execute standard floating-point instructions. Floating-point representation permits a greater range of number values than is possible with a 32-bit integer. The FP750 is a hex-height module and is easily installed in the CPU backplane. It is functionally transparent to the user and hardware and software modifications are not required. The FP750 receives an opcode from the CPU and decodes the information into a starting microaddress. The outputs of the control store ROM control the arithmetic operations and data path logic. The FP750 executes addition, subtraction, multiplication, and division instructions, which operate on single-precision (32-bit) and double-precision (64-bit) data. It executes extended multiply (EMOD) and polynomial evaluation (POLY) instructions, and converts the data between integer and floating-point formats and between single- and double-precision floating-point formats. The FP750 also executes integer multiplication instruction. The floating-point instructions performed by the FP750 are listed in table 4-1.

Table 4-1 ▪ FPA Floating, Double, and Integer Instructions

Opcode (Hexadecimal)	Mnenomic	Opcode (Hexadecimal)	Mnemonic
40	ADDF2	62	SUBD2
41	ADDF3	63	SUBD3
42	SUBF2	64	MULD2
43	SUBF3	65	MULD3
44	MULF2	66	DIVD2
45	MULF3	67	DIVD3
46	DIVD2	68	CVTDB
47	DIVF3	69	CVTDW
48	CVTFB	6A	CVTDL
4A	CVTFL	6B	CVTRDL
4B	CVTRFL	6C	CVTBD
4C	CVTBF	6D	CVTWD
4D	CVTWF	6E	CVTLD
4E	CVTLF	71	CMPD
51	CMPF	74	EMODD
54	EMODF	76	CVTDF
55	POLYF	7A	EMUL
56	CVTFD	A4	MULW3
84	MULB2	A5	MULW3
85	MULB3	C4	MULL2
60	ADDD2	C5	MULL3
61	ADDD3		

Chapter 5 · VAX-11/780 and VAX-11/785 Processors

The VAX-11/780 and VAX-11/785 processors are high-end members of the VAX processor family. The VAX-11/780, the original high-performance VAX member, has become an industry standard. The VAX-11/785 processor includes the same features as the VAX-11/780 but performance has been increased. Both the VAX-11/780 and the VAX-11/785 are microprogrammed, 32-bit processors that operate with the VMS operating system or with ULTRIX-32 operating system software. The VMS system provides a reliable, high-performance multiuser environment for time-sharing functions, for batch processing, and for realtime applications. The ULTRIX-32 system, a virtual-memory operating system with complete UNIX functionality, includes demand paging and enhanced performance for applications that require large memory storage. Figure 5-1 shows the VAX-11/780 processor unit. The VAX-11/780 and VAX-11/785 processor cabinets are physically similar except for the name plate designations.

The central processing unit (CPU) performs the logic and arithmetic operations and controls the transfer of information to devices. It executes a large set of variable-length instructions in native mode, and nonprivileged PDP-11 instructions in compatibility mode. The 32-bit addressing and data capability of the processors permits direct access to four billion bytes of virtual address space. The processor's memory management hardware translates a virtual address to a physical address under operating system control. The processor offers a variety of addressing modes that use the general purpose registers to identify instruction operand locations, including an indexed addressing mode that provides true post-indexing capability.

The processor's instruction set includes integral decimal, character string, and floating-point instructions, as well as integer, logical, and bit field instructions. Floating-point instruction execution can be enhanced by an optional floating-point accelerator.

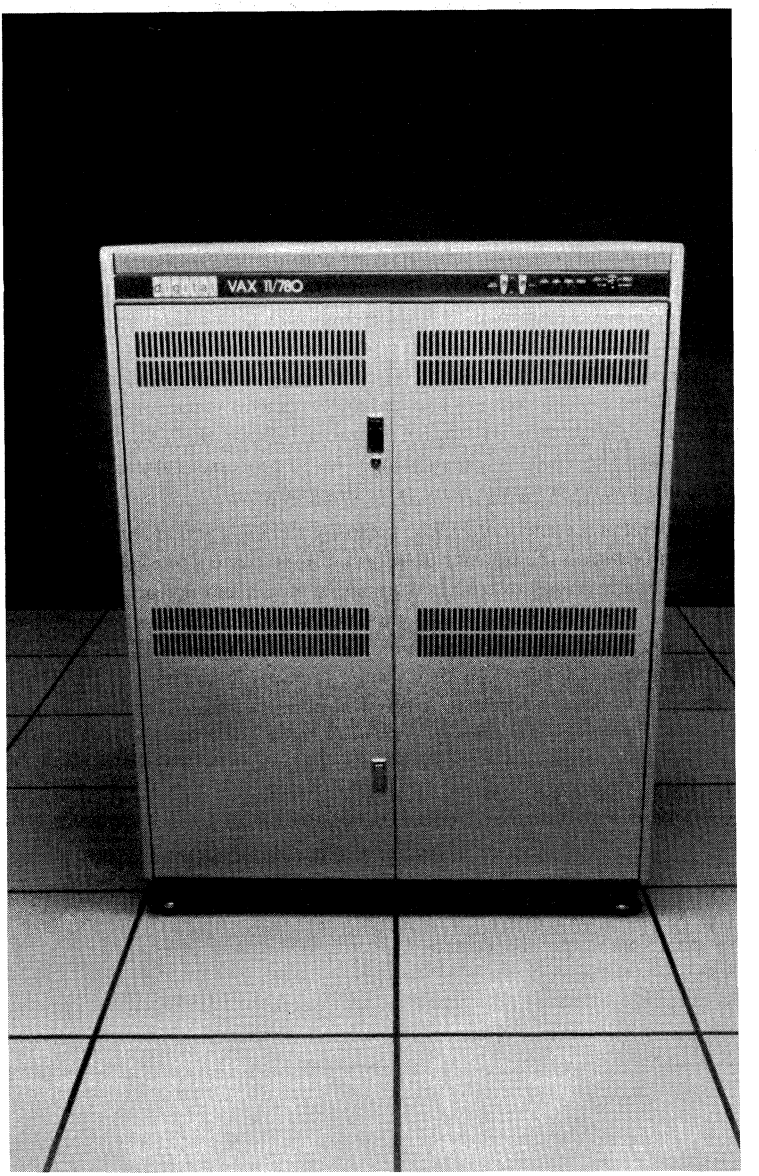


Figure 5-1 ▪ VAX-11/780 Processor Unit

• VAX-11/780 Processor System

The VAX-11/780 processor is the lowest priced VAX processor that allows the full implementation of I/O subsystems and devices available for the VAX processor family. It offers the reliability and performance that has been established and proven over many years in a variety of applications. The continued developments in I/O interfaces and devices, mass storage facilities, VAXcluster configurations, and networking applications have enhanced the VAX-11/780 system performance.

The VAX-11/780 processor includes virtual memory management, bootstrap loader, standard instructions for packed decimal, floating- and fixed-point arithmetic functions, and character and string manipulations. It also contains 8 Kbytes of bipolar cache memory with parity, a high-precision programmable realtime clock, a time-of-year clock with battery backup, and 2 Kwords (99-bit words) of writable-control-store memory. The CPU contains a console subsystem that includes the RX01 floppy-disk drive and an LSI-11 microcomputer to control the console subsystem operation. Some of the processor features are:

- 8-Kbyte, two-way associative cache memory improves main memory access time to increase overall system performance.
- Compatibility mode allows the programs developed for the PDP-11 to operate with the VAX processors.
- 2 Kwords of writable-control store increases throughput by allowing the user to create routines in microcode tailored for specific applications.
- Optional high-performance floating-point accelerator decreases instruction execution time of floating-point arithmetic and some integer arithmetic operations.
- CPU memory can expand up to 32 Mbytes.
- Up to 8 Mbytes of multiport shared memory is available.

• VAX-11/785 Processor

The VAX-11/785 is an increased-performance processor that operates more than 50 percent faster than the VAX-11/780 while using the same power and packaging. The VAX-11/785 supports the same I/O bus structures as the VAX-11/780, thereby allowing the standard VAX interfaces and devices to operate with the system. Because of its increased speed, most of the performance improvement is in CPU processing operations; however, the performance of I/O-intensive applications is also improved because of faster interrupt response time.

The VAX-11/785 processor uses advanced Schottky technology to decrease the logic switching time and includes a cycle time that is 50 percent faster than the VAX-11/780 CPU. The accelerated cycle time results in higher data throughput and faster response time, and provides the ability to support additional users.

The cache memory of the VAX-11/785 processor has been increased to 32 Kbytes from the 8 Kbytes included with the VAX-11/780 processor. This allows the cache to store more data, thereby increasing the probability that the data required by the processor will be found in cache memory.

The microcode of the VAX-11/785 processor is stored in RAMs, which increases the performance of the CPU and allows revisions to be easily incorporated.

The console memory in the VAX-11/785 processor provides 48 Kbytes of storage, compared to the 16 Kbytes of memory included with the VAX-11/780. The increased memory size improves the console subsystem performance and allows the RAM-based microcode to be easily implemented.

The VAX-11/785 processor includes virtual memory management, bootstrap loader, standard instructions for packed decimal, floating-point (G and H data types) and fixed-point arithmetic functions, and character and string manipulations. It also contains 32-Kbyte, two-way set-associative cache memory with parity, a high-precision programmable realtime clock, a time-of-year clock with battery backup, and 8 Kwords (99-bit words) of writable-control-store memory. The CPU contains a console subsystem that includes the RX01 floppy-disk drive and an LSI-11 microcomputer to control the console subsystem operation. Some of the processor features are

- 32-Kbyte cache memory improves main memory access time to increase overall system performance.
- Compatibility mode allows the programs developed for PDP-11 to operate with the VAX processors.
- 8 Kbytes of writable-control store increases throughput by allowing the user to create routines in microcode for specific applications.
- Optional high-performance floating-point accelerator decreases instruction execution time of floating-point arithmetic and some integer arithmetic operations.
- CPU memory can expand up to 32 Mbytes.
- Up to 8 Mbytes of multiport shared memory is available.

• VAX-11/780 and VAX-11/785 System Configuration

The standard components included with the VAX-11/780 and VAX-11/785 processors are

- CPU cabinet and power supplies
- 2 Mbytes of ECC MOS memory
- RX01 floppy disk console subsystem
- PDP-11/03 console microprocessor
- H9652 UNIBUS expansion cabinet
- VAX/VMS license and warranty or ULTRIX-32 user license for up to 32 users

The VAX-11/780 and VAX-11/785 processors are contained in a cabinet 153.0 centimeters (60.25 inches) high and 181.1 centimeters (46.5 inches) wide. The cabinet, shown in figure 5-2, is on casters. Double doors on the front and rear of the cabinet allow easy access to the processor components. The right side of the unit contains the memory controller and memory modules, two option panel spaces for mounting adapters, the memory power supply, and the RX01 floppy disk subsystem. The CPU modules are mounted in the backplane located on the left side of the unit, which also contains space for mounting three system unit (SU) backplanes. The left side contains the console interface, backplane and modules for the DW780 UNIBUS adapter, and reserved space for the FP780 (VAX-11/780) or FP785 (VAX-11/785) floating-point accelerator. Two power supplies are included for the CPU and space is available for mounting the power supply provided with the optional floating-point accelerator. Blower units located in both sides of the cabinet provide cooling for the logic and power supplies.

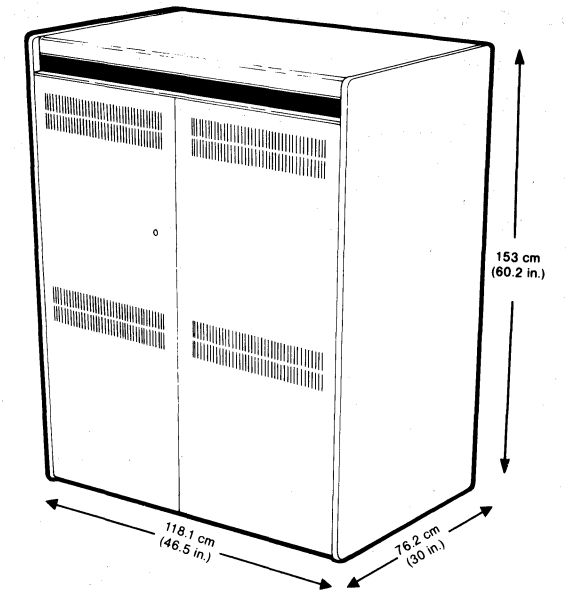


Figure 5-2 ▪ VAX-11/780 and VAX-11/785 Processor Configuration

UNIBUS and CPU Expansion Cabinets

The H9652-M UNIBUS expansion cabinet allows the UNIBUS to be extended from the CPU cabinet. The cabinet, shown in figure 5-3, is 153.0 centimeters (60.25 inches) high and 69.21 centimeters (27.25 inches) wide and contains castors and leveling feet. The H9652-M cabinet is included with the VAX-11/780 and VAX-11/785 system. It contains a BA11-K expansion box and space for mounting a second BA11-K box. The BA11-K box that is included contains a DD11-DK UNIBUS backplane for UNIBUS interface and controllers. Two disk drive units, each with vertical height of 26.67 centimeters (10.5 inches), can be installed in the cabinet if the second BA11-K unit is not included. The DD11-DK backplane provides space for mounting seven hex-height modules and two quad-height modules. The connector panels at the rear of the cabinet can be removed when device connectors are installed.

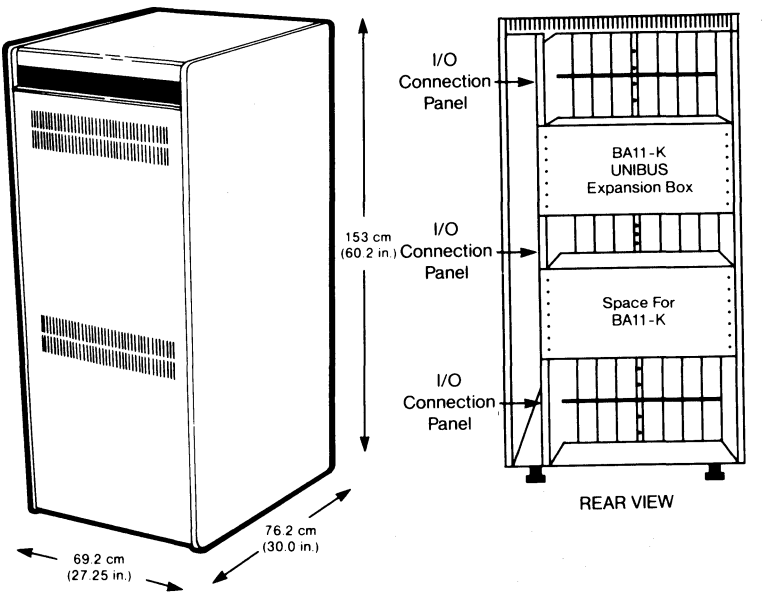


Figure 5-3 • UNIBUS Expansion Cabinet Configuration (H9652-M)

The H9652-H CPU expansion cabinet is used to extend the synchronous backplane interconnect (SBI) bus from the CPU cabinet. The cabinet, shown in figure 5-4, is 153.0 centimeters (60.25 inches) high and 69.21 centimeters (27.25 inches) wide and contains castors and leveling feet. The H9652-H cabinet provides space for four SBI options panels. The options that can be installed are memory modules, DW780 UNIBUS adapter modules, DR780 DDI adapter modules, RH780 MASSBUS adapter modules, and CI780 Computer Interconnect adapter modules. The cabinet includes a power supply and provides space for two additional power supplies, for the H7112 memory battery backup unit, and for a power control panel.

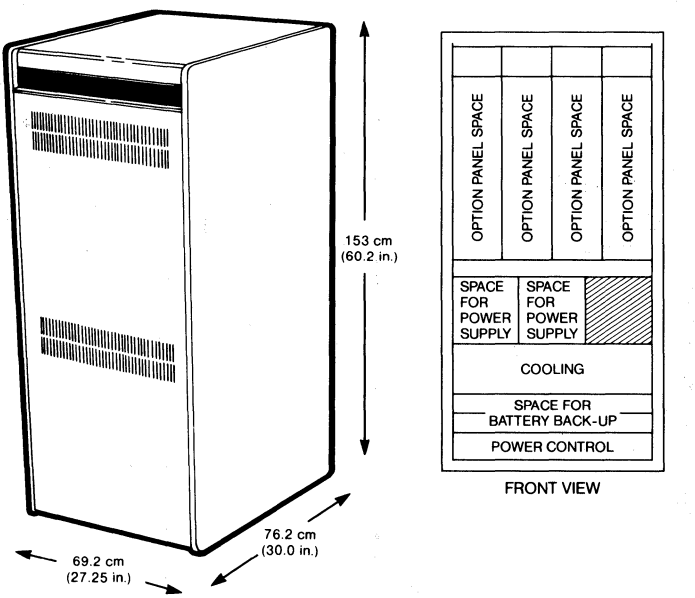


Figure 5-4 ▪ CPU Expansion Cabinet Configuration (H9652-H)

VAX-11/780 and VAX-11/785 Processor Organization

The VAX-11/780 and VAX-11/785 processors each consist of 22 hex-height modules that perform the arithmetic and logical operations of the CPU and control the transfer of information to the system devices. The floating-point accelerator option is contained on five additional hex-height modules. The optional control store logic is contained on a single hex-height module. Figure 5-5 shows the processor components and the interconnecting bus structure.

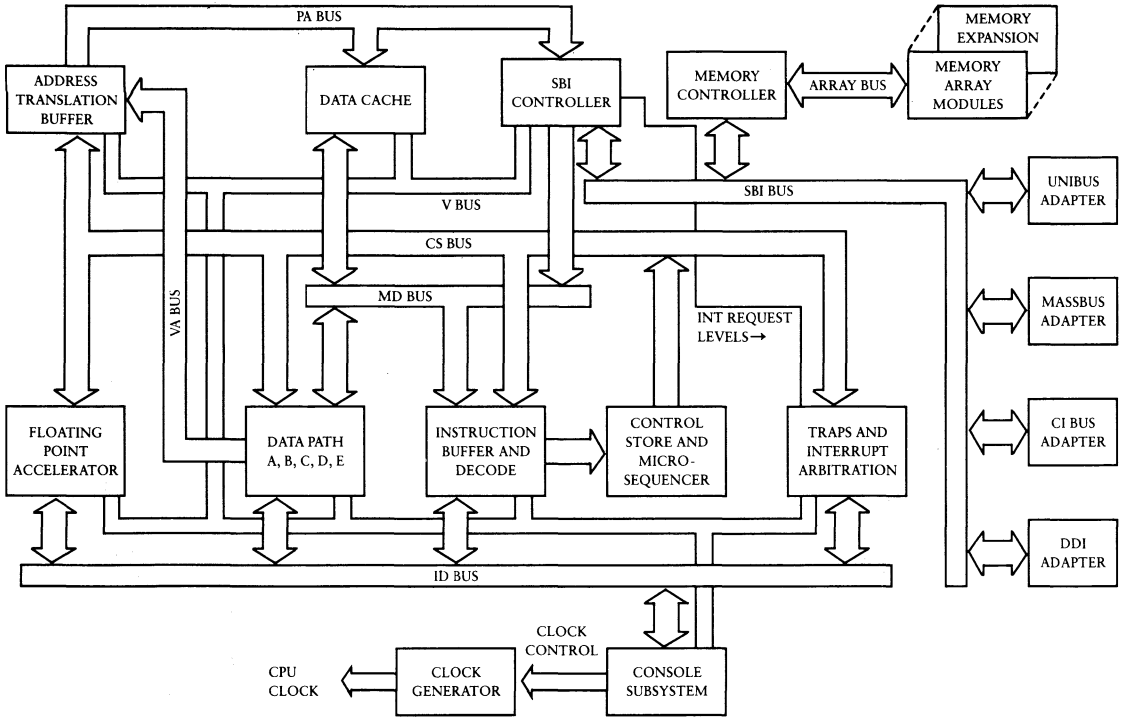


Figure 5-5 • VAX-11/780 and VAX-11/785 Processor Components

- CONSOLE SUBSYSTEM

The console subsystem controls the communication between the CPU and the console terminal, RX01 disk drive, control panel, and remote diagnostic port. The console subsystem, shown in figure 5-6, consists of an LSI-11 microprocessor and RAM memory, a console/CPU interface, a DLV11-E console terminal interface, an RX01 floppy disk drive and RXV11 drive interface. An optional DLV11-E forms the diagnostic port interface. The console subsystem monitors and controls the switches and indicators of the control panel.

The LSI-11 microprocessor and memory control the console subsystem operation. Communication between the LSI-11 microprocessor and subsystem components is through 16 lines that share both address information and data and 14 control signal lines of the Q-bus.

The RX01 floppy disk drive is an inexpensive and reliable device that stores microdiagnostics and system software initiated both locally and remotely. It simplifies system bootstrapping and initialization, and improves software update distribution. The unattended restart feature of the CPU enables the system to restart or reboot itself from the RX01 after a recovery from a power failure or system halt condition.

The main communication path between the console subsystem components and the CPU is through the console/CPU interface module. The interface contains a 4K by 16-bit ROM that contains the core of the LSI-11 console operating system, including the power up routines and the terminal and floppy disk drivers. The console interface module communicates with the CPU through the internal data (ID) bus and through the diagnostic (V) bus. The console interface provides the clock control signals to the processor clock module to synchronize the processor operations and serves as a time base for processing functions. The LSI-11 begins executing instructions in the ROM when power is applied to the system. It contains the interfaces for the console subsystem bus structures, including the registers accessible to each bus, the hardware necessary to implement the console functions, and a 4K by 16-bit ROM, which provides the core of the console LSI-11 software.

The DLV11-E is an EIA communications interface that allows industry-compatible communications.

Switches and indicators on the control panel monitor and select the system operations through the console subsystem.

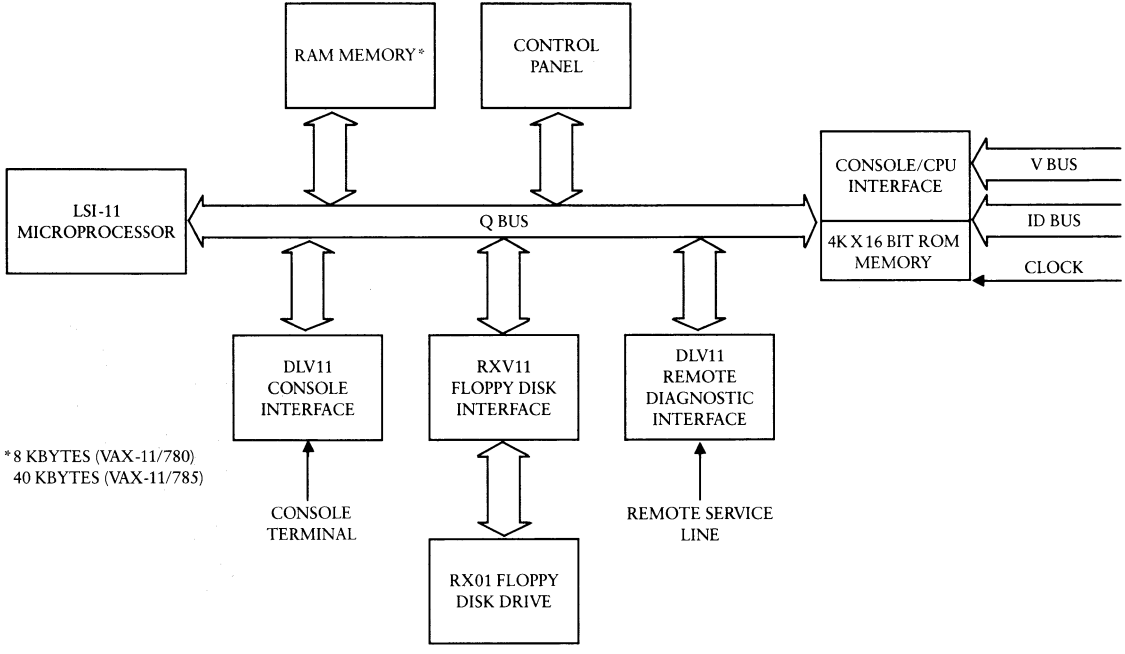


Figure 5-6 • Console Subsystem Components

▪ INTERNAL BUS STRUCTURE

The internal data bus (ID) is a high-speed data path that connects the console subsystem to the major functional areas of the central processor. It transfers data between the instruction buffer data paths, floating-point accelerator, and console subsystem. The ID bus is controlled by the console interface logic during maintenance operations and allows access to writable-control-store information and information in internal registers. The ID bus consists of 32 data lines, six address lines, and one write control line. The address lines specify the internal register that has been selected as the source or destination of the data. The write control line indicates whether the internal register is to receive or transmit the data.

The visibility (V) bus is used to isolate CPU failures during diagnostic maintenance. The V bus consists of eight serial-data lines, a load signal line, a clock signal line, and a self-test line. Each of the CPU modules that connect to the V bus contains one or more V bus shift registers. The data input lines to the shift registers monitor test points on the modules. The test point information is loaded in parallel into the shift register and the clock signal transfers the data serially onto the V bus and into a console register, where it can be monitored by the console software.

The physical address (PA) bus is a 28-bit bidirectional bus that transfers the translated physical address from the translation buffer to the cache memory and the SBI controller.

The synchronous backplane interconnect bus is a bidirectional information path that supports the communication protocol for the data exchanges between the CPU, memory controller, and I/O adapters. The SBI bus provides parallel information exchanges that are checked and synchronized with the system clock.

The control store (CS) bus provides the path for the transfer of each microword field to the CPU logic areas. The CS bus consists of 96 data bits and three parity bits, one for each 32-bit data segment. The microword is grouped into fields, each of which provides control of an area of the processor.

The memory data (MD) bus is a bidirectional path for longword-aligned data exchanges. The MD bus connects the data path of the CPU and the instruction buffer to the cache and SBI controller. The bus consists of 32 data lines, four parity lines, and four mask lines. The parity lines provide the parity for each of the four data bytes. The mask bits are also associated with each of the four bytes and indicate which bytes are to be read or written.

- PROCESSOR LOGIC ELEMENTS

The VAX-11/780 and VAX-11/785 processors consist of the following logic elements:

- A control store that contains the VAX-11 instruction set, writable control store, and optional user control store information.
- A microsequencer that generates the microword address.
- A data path that provides the arithmetic and logic functions of the processor.
- An address translation buffer that contains frequently used virtual-to-physical address translations.
- A primary cache memory for the data transferred from memory.
- An instruction buffer that decodes and stores instructions for evaluation by the processor.
- A processor clock that provides the system timing functions.
- An interrupt control that evaluates interrupts and exceptions.

Control Store Functions–The basic microprogram of the VAX-11/780 is contained in a 4K by 99-bit PROM control store (PCS). The system also includes a 2K by 99-bit writable diagnostic control store (WDCS) that contains diagnostic microprogram routines and updates to the basic microprogram. In the VAX-11/785 the microcode is stored in eight 256K by 99-bit ROMs, allowing the microcode to be easily modified and increasing the performance of the CPU. The microcode controls the operation and sequencing of the processor and includes the native mode, PDP-11 compatibility mode, and floating-point instruction sets. A microword consists of 96 data bits and 3 parity bits, one for each 32-bit segment. It is addressed by the microsequencer or the instruction decode logic. The control store also contains a 96-bit buffer that enables it to execute one microword while simultaneously fetching the next. Parity is checked on each microword read from the PCS or WDCS, and one parity bit corresponds to each 32-bit section.

Microsequencer Functions–The microsequencer contains the logic required to generate the microword address. The microsequencer monitors the console operations and processor conditions, selects the proper address for the control store word, and transfers the address to the PCS or WDCS. The microsequencer controls the entry into the microprogram during normal program flow and during the powerup and powerdown sequence, console operations, microtraps, changes in the microcode, and program stalls.

Data Path Functions—The data path logic contains four functional sections: address, arithmetic, data, and exponent. Each section operates independently and is capable of processing data and addresses in parallel with operations being performed in other sections. The data path operations are defined by the 96-bit words of the microprogram. The execution of each microinstruction causes a sequence of operations to be performed in the data path logic.

The arithmetic section performs arithmetic and logic operations, bit mask and constant generation, and data shifting operations. It also provides the temporary storage for data and addresses. The data and address information is transferred between other sections of the data paths through the arithmetic logic unit (ALU) of the arithmetic section.

The address section contains the virtual address (VA) register, instruction buffer address register (VIBA), and the program counter (PC). The VA register stores the virtual address that is to be translated into a physical address of a memory location. The VIBA stores the address of the instruction stream data that will be loaded into the instruction buffer. The PC stores the address of the instruction opcode each time a new instruction sequence is started.

The data-shift-and-rotate section packs, unpacks floating-point and decimal string data, and shifts and byte aligns the operand data. The data section contains two 32-bit registers that are used as temporary storage of operand data. It is an interface for the transfer of data to and from memory through the memory data bus and between the internal registers through the ID bus.

The exponent and sign section processes the exponent value of the floating-point numbers while the processing of the fractions is performed in the arithmetic and data sections.

Address Translation Buffer—The address translation buffer is a cache of physical address translations. It significantly reduces the amount of time required by the CPU to perform the repetitive tasks of dynamic address translation. In the VAX-11/780 processor, the cache contains 128 virtual-to-physical page address translations, which are divided into equal sections: 64 system-space page translations and 64 process-space page translations. In the VAX-11/785 processor, the cache contains 512 virtual-to-physical address translations, which are divided into 256 system-space translations and 256 process-space translations. Each of these sections is two-way associative. Byte parity is assigned on each entry to increase the integrity of the data.

Instruction Buffer and Decode Functions – The instruction buffer improves the system performance by prefetching data in the instruction stream. Data from memory is continually loaded into the 8-byte instruction buffer to eliminate the CPU time required to perform two memory cycles when the bytes of the instruction stream cross 32-bit longword boundaries. The instruction buffer also processes operand specifiers before execution and then routes them to the CPU. The opcode of the instruction is stored in the first byte and the remainder of the operand specifier and extensions are stored in the remaining bytes. The instruction decode logic evaluates the instruction stream data stored in the first two bytes of the instruction register and generates the lower 8-bits of each successive microaddress.

Interrupt and Exception Functions – Interrupts and exceptions cause the processor to transfer control from the executing program to a routine that can service the interrupt or exception. The interrupt and exception logic is contained on the interrupt control module that receives input information from the CS bus and transfers information through the ID bus. A priority of service is assigned to both interrupts and exceptions and the priority is decoded by the interrupt control module. The priority encoder logic, contained on the interrupt control, selects the highest interrupt level requested and generates an interrupt-level-active code. This code generates an interrupt vector, which is the address in memory used to select the interrupt or exception service routine.

Interrupts are the notification of events that are independent of the current instruction execution and are normally assigned a priority level higher than exception. Interrupt requests can be received from devices, controllers, and the processor.

Exceptions are usually assigned the lowest priority, occur during the execution of an instruction, and are serviced in the context of the process that produced the exception. Exceptions caused by conditions that could result in a system failure, however, are assigned the highest priority.

Cache Memory Functions – The cache memory is the primary cache system for all data coming from main memory, including addresses, address translations, and instructions. The cache is two-way set associative and contains 8 Kbytes in the VAX-11/780 processor and 32 Kbytes in the VAX-11/785 processor. The memory cache reduces the average time the processor waits to receive memory data by reading eight bytes at a time from main memory, and transferring four bytes to the CPU data paths or instruction buffer. Since the remaining four bytes are already available, the memory cache also provides prefetching. The cache memory system carries byte parity for both data and addresses to increase data integrity. Cache locations are allocated when data is read from memory. When both locations of the cache are filled with data, the data in one location is replaced.

Processor Clock Functions—The processor contains a programmable realtime clock and a time-of-year clock. The interval or realtime clock is used by the software to measure small time intervals that are identified by interrupts. It contains a crystal-controlled oscillator with an accuracy of 0.01 percent, and a resolution of one microsecond. The time-of-year clock is used by the software to perform timekeeping functions such as providing the correct time to the system after power failure or system interruption. It provides the powerfail sequencing, generates the SBI timing signals, and distributes and decodes the SBI signals to the processor logic.

- PROCESSOR OPTIONS

The FP780 (VAX-11/780) and FP785 (VAX-11/785) are floating-point accelerator options that operate with the processor to execute addition, subtraction, multiplication, and division instructions. The floating-point accelerators operate on single- and double-precision floating-point operands, including the special EMOD and POLY instructions in both single- and double-precision formats. The floating-point accelerator also enhances the performance of 32-bit integer multiply instructions.

The KU780 user writable-control-store option is available with the VAX-11/780 processor. It provides control store space for the KE780 extended-range floating-point option and for the QE109-CY microprogramming tools option. The KU780 is a hex-height module that contains 2K by 99-bit word locations. Each word consists of 96-bits plus three parity bits. If the KE780 is not included, these locations are available for user microcode.

The KE780 option provides G and H floating-point microcode. It is used for applications that require extended decimal digit precision. The optional G-floating (double precision) and H-floating (quadruple precision) data points provide up to 33-decimal digit accuracy. The KE780 option attaches to the KU780 module.

- MAIN MEMORY SUBSYSTEM

The main memory subsystem of the VAX-11/780 and VAX-11/785 processors consists of dynamic MOS (metal oxide semiconductor) RAM modules, a memory array bus, and memory controller modules. The memory subsystem contains a 20-slot backplane and connects to the SBI bus through an SBI interface. Two types of memory subsystems are available. The MS780-E subsystem contains 64K by 1-bit MOS RAM technology and the MS780-H subsystem contains 256K by 1-bit MOS RAM technology. The MS780-E subsystem enables up to 16 Mbytes of memory to be installed in a single backplane using 1-Mbyte array modules and two controllers. A second backplane can be installed in the H9652-H CPU cabinet to increase the total memory capacity to 32 Mbytes. The MS780-H memory subsystem allows up to 64 Mbytes of memory to be installed in a single backplane using 4-Mbyte array modules and two controllers.

The memory subsystem supports data transfers up to 13.3 Mbytes per second and contains error correction control (ECC) logic. Eight ECC check bits are stored with each 64-bit quadword and are accessed with the data to determine its integrity. The memory is capable of random-access read and write operations to a single 32-bit longword or extended 64-bit quadword. It also is capable of random-access write operations to an arbitrary byte, to series of contiguous bytes, and to a series of noncontiguous bytes. The memory array modules are organized to optimize 8-byte read or write access. The features of the memory subsystem are as follows:

- Memory configurations are expandable.
- ECC logic enhances data availability and reliability by correcting single-bit errors and detecting double-bit errors within the memory system.
- Subsystem permits write operations to any combination of bytes within an aligned longword.
- Interleaving with two controllers improves memory subsystem throughput on the bus.
- Optional battery backup unit preserves data in memory during a power failure.

Memory Interleaving–The memory subsystem is capable of operating with two memory controllers in the interleaving mode or two-way interleaved mode, improving the memory subsystem throughput on the bus.

In a single-memory controller system, the starting address is assigned by the ROM bootstrap. The size of the memory is encoded from the number of memory array modules installed in the backplane. When two memory controllers are used and each contains an equal amount of memory, the memory access time can be decreased by interleaving. Most memory operations are performed on consecutive memory locations. When one controller is accessing data, the other controller is available to decode an address for the next operation. The two memory controllers access alternate quadwords. With an interleaved memory system, both controllers operate with the same array size and starting address.

In the two-way interleaved mode, four controllers can be used. The second interleaved memory system would have a starting address that is one location greater than the address of the first interleaved set.

Memory Controller Functions—The memory controller forms the nexus from main memory to the SBI and buffers commands, addresses, and data to improve memory throughput. The controller examines the command and address lines of the SBI for each SBI cycle. To initiate and complete a memory-write-masked, interlock-write-masked, or extended-write-masked transaction, the controller must recognize a command/address and data in two or three SBI cycles. To perform a read-masked, extended-read, or interlock-read-masked operation, the controller must recognize only an appropriate command/address. The controller provides the necessary timing and control to complete all memory transactions. It informs a commander of a successful write operation, and contends and arbitrates for SBI bus control to transmit information during a read-masked, extended-read, or interlock-read operation. Before the controller initiates a memory cycle operation, it checks for bus transmission parity errors and other fault conditions and reports any of these conditions to the commander, conforming to the SBI protocol.

The memory controller provides powerup initialization logic and refresh control logic for the dynamic MOS memory devices. The dynamic MOS memory cell is a capacitor in which the stored charge represents a data bit. Because the stored charge tends to diminish, each cell requires a refresh cycle every 2 microseconds to retain the charge.

Data transfers to and from main memory are protected by ECC logic, which performs single-bit error detection and correction and double-bit error detection logic to improve system reliability. This is accomplished by storing eight check bits with the 64 data bits in each memory location. Each check bit is generated by checking the parity of selected groups of data bits in a quadword. When parity is checked during a read, an incorrect bit will be detected by the parity checking logic and will develop a unique 8-bit syndrome that will identify the bit in error. Error correction logic may correct the bit in error. There are 72 unique syndromes related to individual bits in the coded quadword.

Error reporting provides an early warning to protect the system from performance degradation. The system error logging feature requires that single-bit errors and uncorrectable errors be tagged and stored in the memory controller during memory read transmission from the memory subsystem. The memory controller also stores syndrome bits for the first memory-read error and error addresses for error logging and servicing. The memory controller retains this information until the first error is serviced. Ten bits in register B are used for ECC diagnostics only.

When a two-controller interleaved memory configuration is used, the memory controllers must have consecutive SBI bus arbitration lines (TR). The interleave bit of the memory configuration register A will be cleared during powerup and must be set in each controller. Each controller must have the same array size and be issued the same starting address.

When two, two-way interleaved memory systems having four controllers are used, the second interleaved memory system is assigned a starting address that is one location above the final address of the first. The proper starting address will be written into the memory configuration register B from the SBI bus.

A 4-Kbyte programmable read-only memory (ROM) used to bootstrap the operating system is on the memory controller. The ROM is assigned a 4-Kbyte I/O address space and can be addressed through the SBI interface during system initialization. The address, timing, and control logic to read the information from the ROM for booting the system also is contained in the subsystem.

- BASIC MEMORY OPERATIONS

The memory subsystem operates synchronously with the SBI clock cycles. The physical address space of the SBI is divided into two equal areas—memory address space and I/O address space. Figure 5-7 shows the SBI physical address space assignments.

The 28-bit SBI longword address field allows access to up to 512 Mbytes of main memory. The hardware supports a maximum of 64 Mbytes of main memory. Physical memory operations are performed when bit 27 of the SBI addresses is zero; I/O operations occur when bit 27 is one. The operation field identifies one of the six transactions to be performed by the memory subsystem: read masked, extended read, interlock read masked, write masked, extended write masked, and interlock write masked.

A write masked operation transfers from 1 to 4 bytes of data to memory and a read masked operation transfers 4 bytes of data from memory. An extended read is executed to transfer 8 bytes of data (2 longwords) from memory to a requesting nexus. An extended-write-masked operation provides a byte-selectable transfer of up to 8 bytes to memory. Interlock-read-masked and interlock-write-masked operations perform a function similar to that of read masked and write masked operations but they also provide process synchronization.

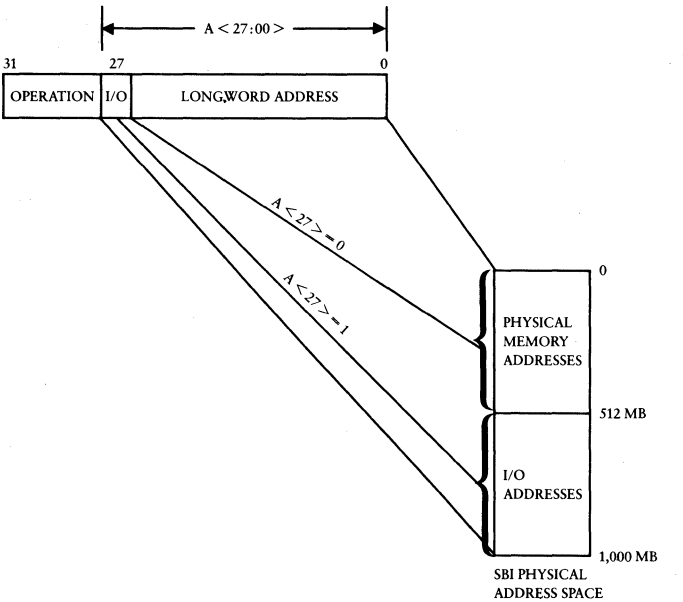


Figure 5-7 ▪ Memory Controller Physical Address Space

During a read cycle, 32 bits of data are accessed from the addressed location in the memory, with the memory controller checking for single- or double-bit errors. Single-bit errors are corrected and rewritten into the memory in a subsequent memory cycle. When a double-bit error is detected, the same data and check code that was read is rewritten to ensure that error will recur on subsequent read operations. An indication of the error condition is tagged and transmitted to the requester with the corrected data or uncorrectable bad data during the next available bus cycle. The data word is transferred with the proper tag and identification (ID) code of the commander that requested the data.

A commander arbitrates and gains control of the bus to initiate a read cycle to a memory subsystem. The commander then transmits a command or address tag and identification code (ID), which is decoded by all the subsystems on the bus. If the address corresponds to the memory subsystem and no faults are detected, the memory subsystem initiates a memory cycle. If a memory cycle is being executed, the command is stored in the queue until the cycle is complete and the memory notifies the commander that the message has been received. The address is fetched from the memory while the memory controller is requesting, arbitrating, and gaining control of the bus for the transfer of data and commander's ID code. Read cycles with single-bit errors require an additional bus cycle to correct the error. In this case, the controller rerequests the bus and transmits the data after gaining control of the bus.

The extended read cycle is similar to the read cycle except that 64 bits of data are accessed from the addressed location. The data is transmitted on the SBI in two successive bus cycles. The lower 32 bits are transmitted before the upper 32 bits. If a single-bit error occurs, the data transmission is delayed until the memory controller again requests the bus and gains control.

The write masked function instructs the addressed memory controller to modify the bytes as specified using data transmitted in the succeeding cycle. This is accomplished by initiating a read cycle followed by a write cycle. During the read portion of the cycle, the 64-bit word is retrieved, the error code is checked, and the appropriate bytes are modified in the upper or lower half of the word. New check bits are then encoded and the modified word is written into the memory. Single-bit errors that occur during the read portion of the cycle are corrected. Uncorrectable errors are rewritten into the memory with an erroneous check code and the new data is not used. Up to four bytes in the upper or lower word can be modified in a write masked cycle.

The extended write masked function instructs the memory controller to first modify the bytes specified by the mask field in the low 32 bits of the addressed storage element, using data transmitted in the succeeding cycle. The controller then modifies the bytes of the high 32-bit storage element as specified by the mask field of the first data-word cycle, using data transmitted in the next cycle. The mask field in the second data word transmission is ignored. This cycle is similar to write masked except that from 1 to 8 bytes of an address can be modified in the upper and lower word. An extended write masked function that specifies modification of all 8 bytes unconditionally writes the new 64 bits and 8 check bits to the designated address.

The interlock cycles, which are special memory cycles used for process synchronization, consist of the interlock read cycle and the interlock write cycle. Each interlocked cycle is a pair of cycles, an interlock read masked followed within an arbitrary number of cycles by an interlock write masked. Interlock read and write cycles are 32-bit operations. An interlock line on the SBI is used to coordinate activity between memory controllers. An interlock timer starts with the acceptance of an interlock write. If the interlock write is not found after 512 bus cycles, the interlock line is cleared.

Implementation of the interlock read masked cycle is similar to that of the read masked cycle, except that a special function code is decoded by the memory controller. The memory controller also monitors the interlock line on the bus for interlock activity. If the interlock line is not asserted, the addressed memory controller acknowledges the cycle and sets its interlock line on the SBI until a valid interlock write has been received. If single-bit errors are detected, the controller corrects the data and transmits the data with the proper tag. If an uncorrectable error occurs, the read data substitute tag and the erroneous data are transmitted and the memory rewrites the erroneous data and ECC. Every commander on the SBI that can issue an interlock command asserts the interlock line on the bus for one cycle immediately following the interlock read mask command. This ensures the cooperation among memory controllers that respond to interlock reads without ambiguity.

The interlock write masked cycle is similar to the write masked cycle except that the interlock line is monitored to verify the integrity of the command before acknowledging and implementing the cycle.

- MEMORY CONFIGURATION REGISTERS

The memory controller contains three memory configuration registers that provide memory configuration and error information to the operating system and diagnostic software. Descriptions of the read-write registers follow.

Memory Configuration Register A—Memory configuration register A contains information on SBI fault status, power status, and memory configuration. The format of the register information is shown in figure 5-8.

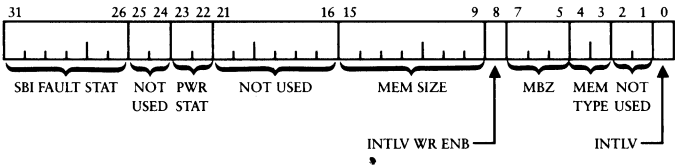


Figure 5-8 • Memory Configuration Register A Format

Memory Configuration Register B—Memory configuration register B provides file pointer, starting address, and ECC control information. The information format is shown in figure 5-9.

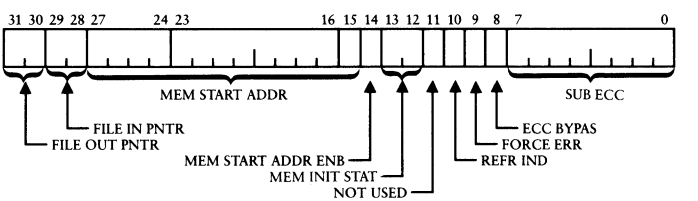


Figure 5-9 • Memory Configuration Register B Format

Memory Configuration Register C—Memory configuration register C contains a summary of the ECC error information. The register format is shown in figure 5-10.

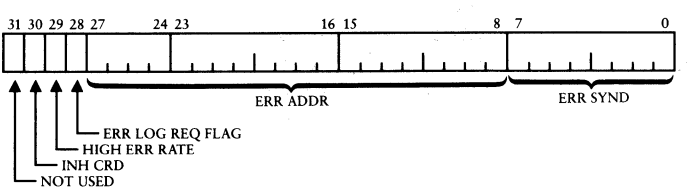


Figure 5-10 • Memory Configuration Register C Format

Shared Multiport Memory

The MA780 multiport memory option is a shared memory subsystem that enables up to four VAX-11/780 or VAX-11/785 processors to access a common bank of ECC MOS memory. The information stored in the multiport memory is accessed at random by any of the processors the same way as a single processor accesses its local memory. Two MA780 subsystems can be connected to each processor and each subsystem can contain up to 2 Mbytes of multiport memory. The features of the MA780 shared memory system are:

- Fast interprocessor communications
- Expanded physical address space up to a system total of 12 Mbytes
- Parallel or sequential MA780 configurations allow exceptionally fast processing
- Access to shared memory that is transparent to the user
- Invalidation logic that permits access of shared memory data through cache memory
- Memory interlock capability to prevent shared memory from being read by one processor while being updated by another processor
- Individual memory port control that permits high overall system uptime

The VMS operating system fully supports MA780 configured systems. Applications built around multiple cooperating processes can be reconfigured to run on a multi-CPU system without program modification. Processes in shared memory can be moved from one processor to another without affecting the programs involved. The VMS system provides support for MA780 configurations in the areas of interprocessor communications, including the sharing of data regions and code, VMS mailboxes, and common event flags. Each CPU in the multiport system operates independently using its own copy of the VMS operating system stored in its local memory.

▪ MA780 SHARED MEMORY CONFIGURATION

The MA780 multiport memory can be mounted in an existing CPU expansion cabinet or is available in a separate cabinet that attaches to the CPU cabinet. The cabinet of the attached version is the same size as the H9652-H CPU expansion cabinet previously described. It includes a port interface, memory controller, and power supply, and has space for up to 2 Mbytes of memory and a memory battery backup unit. Figure 5-11 shows a system configuration using the MA780 shared memory subsystem.

Figure 5-11 • MA780 Dual Processor System Configuration

The MA780 includes 256 Kbytes of ECC MOS memory with the initial configuration. Additional memory can be added in 256-Kbyte increments up to a maximum of 2 Mbytes. The VAX-11/780 system will support two MA780 subsystems, thus increasing the total addressable physical memory to 68 Mbytes per system.

The MA780 shared memory includes two interface ports to allow communication between a VAX-11/780 or VAX-11/785 processor and a second processor. Additional interface port options that mount in the MA780 permit access to a third and fourth CPU. The port options mount in the MA780 subsystem. Systems that contain three or four MA780 interface ports should also include a selective cache invalidate option.

The MA780 cabinet contains power supplies, cables, and memory array modules as shown in figure 5-12. Adequate space is provided in the cabinet to accommodate a second MA780. The memory battery backup option can be installed in the MA780 cabinet and is capable of supporting 2 Mbytes of memory for a minimum of 10 minutes. Smaller amounts of memory are supported for longer periods of time.

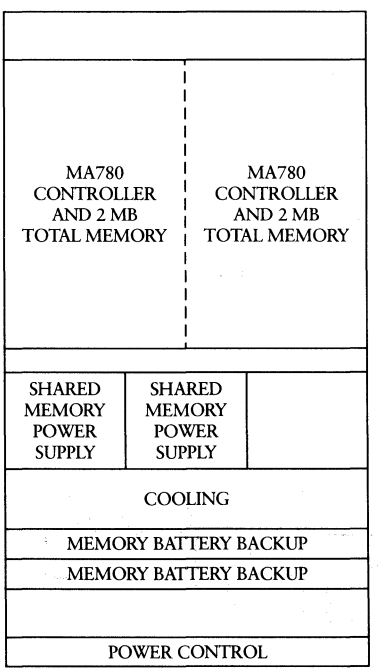


Figure 5-12 ▪ MA780 Shared Memory Cabinet Configuration

- SHARED MEMORY OPERATION

The MA780 has a maximum throughput rate of 11 Mbytes per second for 64-bit quadword (8-byte) data transfers. Smaller sized transfers result in lower throughput. The throughput rate is a function of the number of cache hits and the I/O and SBI traffic. The high throughput rate is achieved through the use of a separate buffer in each interface port for commands and data. Each port is serviced on a demand basis that allows first-in requests to be serviced first; therefore, inactive ports do not have to be polled. A port gains access to the shared memory within three memory cycles.

The MA780 allows parallel or sequential (pipeline) processing. In parallel processing, a task is shared between two or more processors, thereby reducing the total processing time. Sequential processing can increase the system throughput by allowing data to be exchanged between processors that are handling sequential parts of an application. Each processor is dedicated to a specific portion of the total application and, upon completion of that portion, passes the results to the next processor. Figure 5-13 shows the parallel and sequential processing configurations.

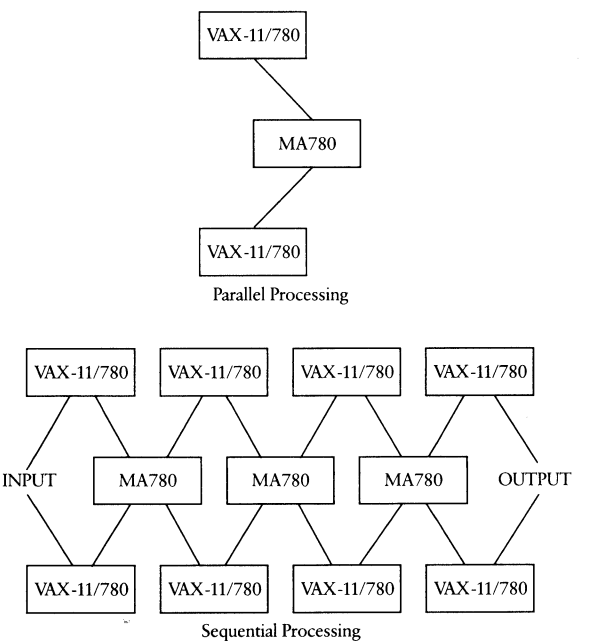


Figure 5-13 • MA780 Parallel and Sequential Processing Configuration

The shared memory includes error correcting code (ECC) to correct single-bit errors and detect double-bit errors during memory read and write transactions. Parity error checking is included on internal buses.

During a processor transaction with shared memory, all other processors are prevented from accessing the memory until the first processor has completed its transaction. After data has been written to a shared memory location, the MA780 sends an invalidate signal and address to each processor. The processor checks the cache to determine if that address is present and marks the contents as invalid. During subsequent read operations, the processor is required to access the new information from shared memory.

A selective cache invalidation option is available for multiprocessor systems that maintain a high rate of SBI transactions. In systems with three or four processors, the MA780 sends invalidate signals only to the processors that have accessed a given location in shared memory. The invalidation map has one entry for each 64-bit quadword in shared memory. This entry contains four bits, one for each processor. When a processor reads a 64-bit quadword, the MA780 controller sets the bit in the correct invalidation map entry. When a processor writes into that 64-bit quadword again, the MA780 controller notifies the processors whose bits have been set in the invalidation map. The bits are then cleared, except in the processor that is performing the write operation. Overall traffic on all SBIs and the cache activity is therefore minimized while data integrity is maintained.

The MA780 supports a failsoft capability, which is the ability of the system to operate when certain elements in the system are failing. The system diagnostics can access the MA780 through any one of its memory access ports, and any processor connected in the system can read the multiport status register. Each port can be separately controlled and a defective processor can be disabled. This permits the system to be serviced when an interface port or processor is malfunctioning. The MA780 generates interrupts to notify a port of a power failure or an error condition. The VMS operating system logs the detected errors for evaluation.

Chapter 6 · VAX-11/782 Attached-processor System

The VAX-11/782 attached-processor system is a high-end member of the VAX family and is software-compatible with other VAX systems because of the VAX architecture. Its tightly coupled, asymmetrical, dual-processor system is based on the MA780 shared-memory subsystem. The system contains two VAX-11/780 processors: a primary processor and a secondary (attached) processor. Each processor shares the same data structures and up to 8 Mbytes of data in the MA780 multiport memory. All kernel-mode, interrupt-code, and I/O operations are executed by the primary processor, which also schedules the operations of the attached processor. The attached processor communicates with the primary processor through the shared memory and provides additional computational power. Because the attached processor is free from I/O activities, its time can be scheduled entirely for data processing activities. The processors execute the same copy of the VAX/VMS operating system code; therefore, only one copy of the operating system has to be maintained.

A VAX-11/782 upgrade package (782UP-F) is available to enable owners of VAX-11/780 processors to convert their system to the higher performance VAX-11/782 dual-processor configuration.

Figure 6-1 shows the system configuration of a typical VAX-11/782 attached-processor system.

The VAX-11/782 system provides up to an 80 percent performance improvement over the VAX-11/780 single processor system. It is suitable for both commercial and technical applications, including experimental data reduction, structural analysis, electronic and mechanical design with interactive graphics, high-energy physics, and quantum mechanics research. With the VAX-11/782 system, many users have access to large-scale databases for econometric modeling, forecasts and business simulation, and statistical processing applications.

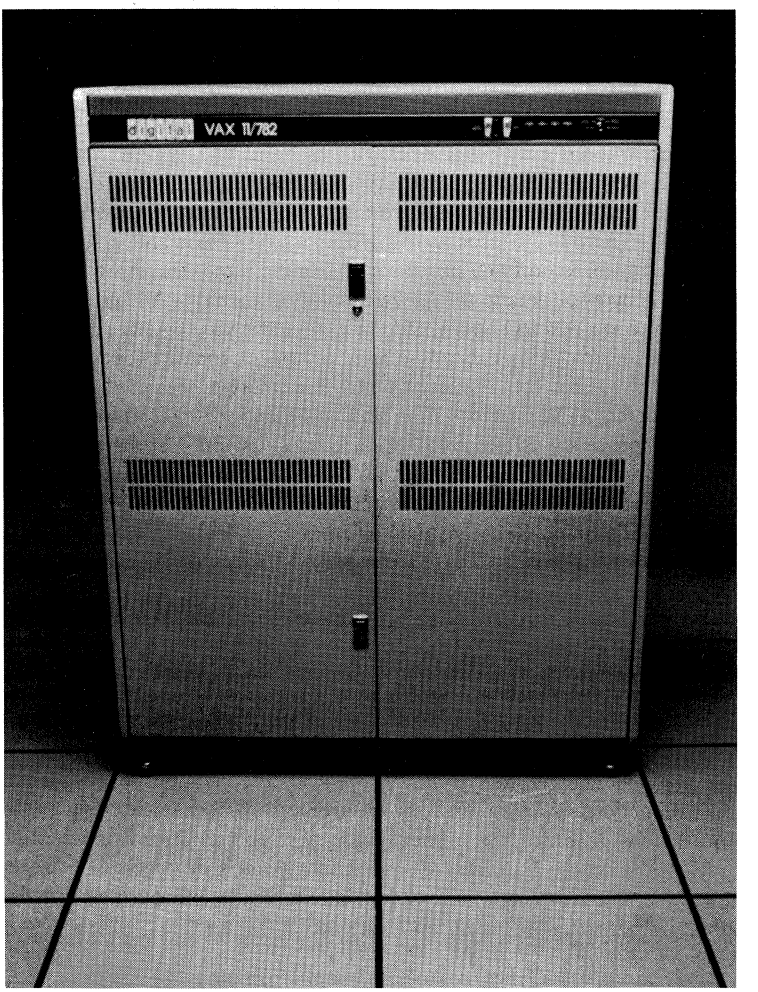


Figure 6-1 ▪ VAX-11/782 Attached-processor System

The VAX/VMS operating system scheduler dynamically allocates compute-intensive functions to the attached processor. Functions that require I/O transfers are assigned to the primary processor. The VAX/VMS software system supports the operation of MA780 subsystems, and applications developed for individual VAX systems will operate on the VAX-11/782 system with no program modification. Some of the system features are:

- As a high-performance VAX system, it can run demanding applications.
- Existing VAX-11/780 systems can be upgraded to VAX-11/782 systems.
- Up to 8 Mbytes of main memory can be shared between processors.
- Processor interaction is transparent to the user.
- Improved dynamic load leveling provides optimal use of system resources.
- Applications can be transferred from a single processor to an attached-processor system without modification.
- Only one copy of the VMS operating system has to be maintained.

VAX-11/782 Processor System Configuration

The VAX-11/782 attached-processor system consists of two VAX-11/780 processors and cabinets, one MA780 shared-memory subsystem cabinet, and one UNIBUS expansion cabinet. The primary processor includes the VAX-11/780 CPU local memory for diagnostics programs only, a shared-memory interface, and a DW780 UNIBUS adapter. All I/O devices connect to the primary processor. The secondary (attached) processor includes a VAX-11/780 CPU, local memory for diagnostic programs, and a shared-memory interface. The shared-memory cabinet in the basic system contains 2 Mbytes of ECC MOS memory and a memory battery backup unit. The same version of the VAX/VMS operating system and microcode is used for each processor and is included with the basic system. Figure 6-2 shows the system configuration.

The MA780 cabinet contains the H7112 memory battery backup units that maintain data in memory during a power failure. Processor options installed in the primary processor, such as the floating-point accelerator, must also be installed in the attached processor.

Communication between the processors and shared memory is through the port interfaces, which are in the processor SBI backplane. The interfaces connect by cable to the MA780 shared memory unit. Figure 6-3 shows the VAX-11/780 system components and bus configuration.

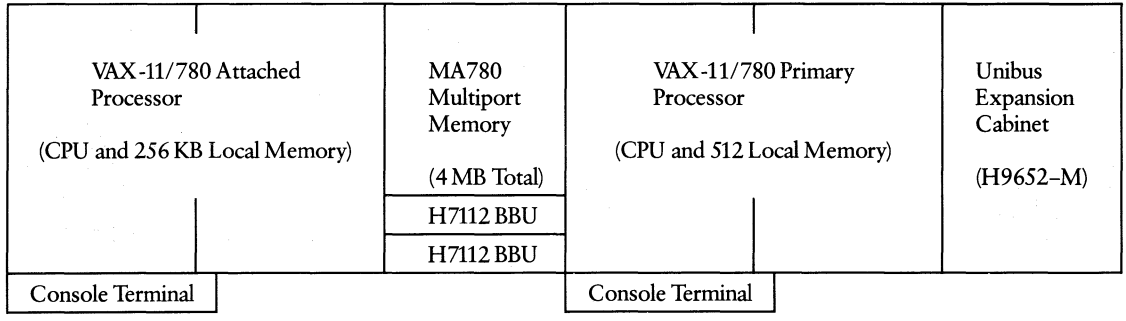


Figure 6-2 ▪ VAX-11/782 Attached-processor System Configuration

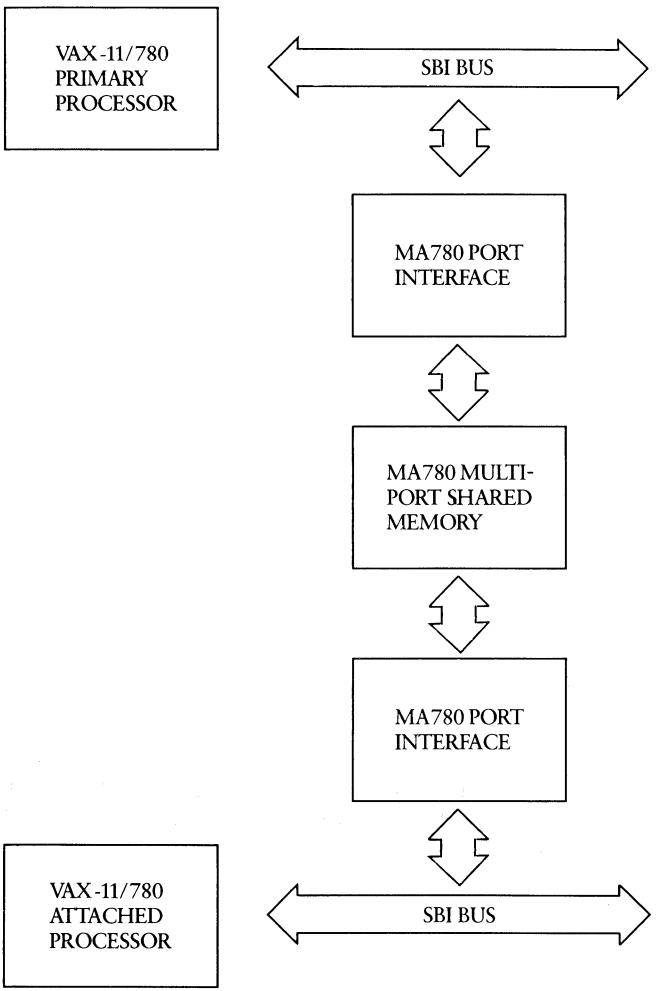


Figure 6-3 ▪ VAX-11/782 System Components

VAX-11/782 Upgrade Package

Single VAX-11/780 processor systems can be converted to a VAX-11/782 attached-processor system by adding the VAX-11/782 upgrade package. This option includes a VAX-11/780 processor, MA780 multiport memory subsystem, cache invalidate option, 1 Mbyte of ECC MOS memory, and LA120 console terminal. If the single VAX-11/780 processor contains an FP780 floating-point accelerator option or a KE780 extended range G and H floating-point data type option, the attached processor must contain the same options. The multiport memory subsystem includes two port interfaces that connect to VAX-11/780 processors. Figure 6-4 shows the VAX-11/782 upgrade system configuration.

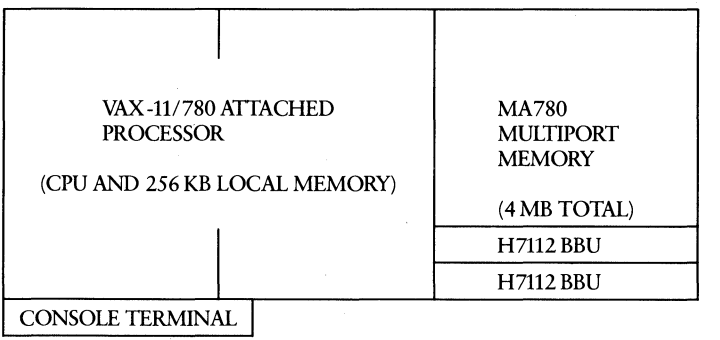


Figure 6-4 ▪ VAX-11/782 Upgrade Package Configuration

VAX-11/782 System Operation

The operation of the VAX-11/780 processor and the MA780 shared memory subsystem is described in "Chapter 5, VAX-11/780 and VAX-11/785 Processors." The VAX-11/782 system provides a maximum throughput rate of 11 Mbytes per second with 64-bit quadword (8-byte) data transfers. The actual throughput rate is dependent on the number of cache hits encounted and the number of I/O-device and SBI-bus transfers in process.

The cache invalidation option (MA780-D), included with the attached-processor system, is used to minimize data transfers and cache activities while maintaining the integrity of the data. The cache invalidate option allows systems with a high rate of transfer on the SBI bus to send invalidate signals to the CPU that has accessed a location in the MA780 multiport memory. An invalidation map contains one entry for each 64-bit quadword in shared memory and an entry contains four bits, one for each CPU. When a CPU reads a 64-bit quadword from shared memory, the MA780 controller sets a bit in the correct invalidation map entry. When a processor writes into that 64-bit quadword, the MA780 controller notifies the processor that has the bits set in the validation map and clears the remaining bits.

Applications designed for multiple cooperating processes can be reconfigured to run on a multiprocessor system without program modification. Processes running on a VAX-11/782 attached-processor system will execute on one processor and then the other, all transparent to the programs involved.

Process scheduling in the VAX-11/780 processor is performed similarly to that of a single VAX-11/780 processor system, with the highest-priority transaction executed first. The attached processor, however, will not be preempted by a higher priority process and will not execute transactions in kernel mode. The primary processor performs the scheduling for the system by scheduling the attached processor first and then scheduling itself.

When a process cannot be performed on the attached processor, the primary will execute a process and cause an asynchronous system trap interrupt to occur when the process leaves kernel mode. The interrupt causes the primary processor to reschedule the process to run on the attached processor.

- PROCESSOR INITIALIZATION

The primary processor is initialized before the attached-processor. A standard bootstrap loading procedure loads the operating system in the primary processor and the multiprocessing code into nonpaged pool. A Digital Command Language (DCL) command loads the multiprocessing code and is usually incorporated into the startup command file at each site by the system manager. A new system control block (SCB) is initialized for the attached processor and the primary SCB is modified to include the multiprocessing scheduling code and MA780 interrupt communication. The operating system is then loaded in the attached processor and it interrupts the primary processor to request a process to execute. The primary processor is responsible for scheduling itself and the attached processor.

▪ ATTACHED PROCESSOR STATES

Figure 6-5 shows the transition states of the attached processor. The attached processor is initialized by the primary processor when the multiprocessing code is loaded. After initialization, the attached processor becomes idle until the next rescheduling operation is performed by the primary processor. The primary processor initiates the scheduling and assigns tasks for the attached processor, which enters the busy state. The attached processor checks the state, performs a load-process context, and sets its state to execute. The attached processor will continue to execute its current process until the time allocated for the process is exceeded or the process requests an action to be performed in kernel mode. When this occurs, the attached processor saves the processor context, enters the drop state, and notifies the primary processor that it is available for rescheduling. After receiving the process from the attached processor, the primary processor sets the attached processor state to idle. The stop state, used to disable the attached processor, is initiated by the system manager with a DCL command or by the primary processor during operations such as bugchecks. The attached processor can be enabled by another DCL command or by rebooting.

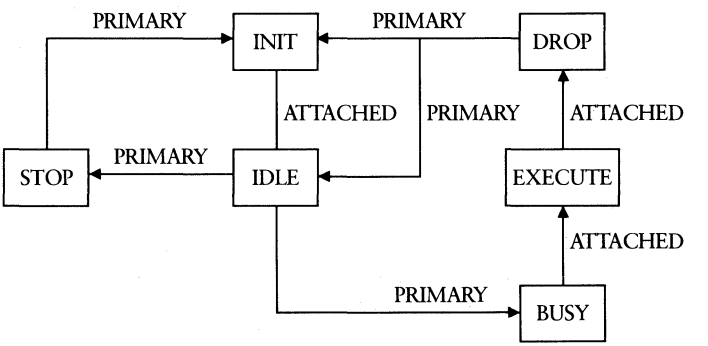


Figure 6-5 ▪ VAX-11/782 Attached Processor States

- MULTIPROCESSING INTERRUPT COMMUNICATIONS

The multiprocessing code uses MA780 interprocessor interrupts. The primary processor interrupts the attached processor to request an invalidate of a system space address, to indicate that an asynchronous system trap has occurred for the process currently being executed in the attached processor, or to request a bugcheck. The attached processor interrupts the primary processor to request a reschedule event, to log an error, or to request a bugcheck.

Many of the fault-handling features of VAX/VMS are included with a VAX-11/782 system. These include a powerfail, bugcheck, machine check, automatic restart, and error logging. When the attached processor loses power, the process that is executing on the attached processor is transferred to the primary processor, where the process continues to run. If the power to the primary processor fails, the attached processor waits for the primary processor to be restarted. A bugcheck can be initiated by either processor, stopping the execution of both processors. The processors will synchronize during the bugcheck and both processors can be set to reboot automatically . Machine checks and error logging are performed the same as with a single VAX-11/780 processor system. The error log entries include the system ID to identify the processor that made the entry.

Chapter 7 · VAX 8600 Processor

The VAX 8600 is a powerful, high-performance computer system designed for engineering, technical, commercial, and special applications. It implements the VAX architecture and is supported by the VMS operating system with its associated layered software products. It also supports the ULTRIX-32 operating system, Digital's native-mode UNIX software, which has been functionally enhanced to operate with VAX virtual-memory computers. The plug-compatible VAX 8600 system can replace most VAX-based systems and can be used as a small mainframe system or as a high-end minicomputer for realtime, interactive, and distributed processing applications. The VMS operating system provides compatibility between the VAX 8600, other VAX processors, and the PDP-11 family of processors, thereby allowing owners of these systems to upgrade to the new, more powerful VAX 8600 processor. The VAX 8600 system incorporates the latest technological advancements and diagnostic functions to provide a fast, efficient, and reliable system for many applications. Figure 7-1 shows the VAX 8600 processor. Some of the features and benefits of the system are

- High performance and increased reliability through the use of ECL gate arrays and advanced processor design.
- Comprehensive diagnostic and maintenance features that ensure reliable operation, system availability, and ease of maintenance.
- Main memory expansion up to 32 Mbytes using 256-Kbit MOS RAM integrated circuits.
- Flexible I/O configurations that permit UNIBUS, MASSBUS, CI bus, and DR32 Digital data-interconnect devices to be implemented easily on new and existing VAX 8600 systems.
- An efficient microprocessor-controlled console subsystem with RL02 removable cartridge disk drive for diagnostics, microcode loading, and automatic remote testing.
- Quiet operation through the use of acoustical mufflers.

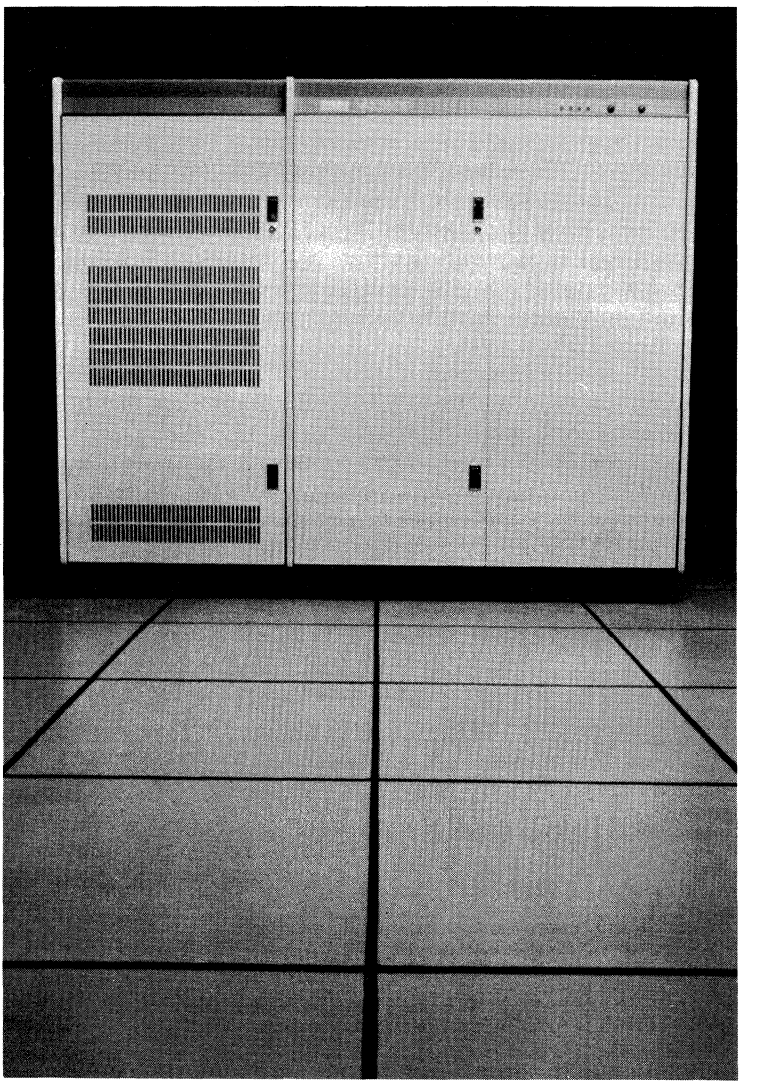


Figure 7-1 ▪ VAX 8600 Processor Unit

· VAX 8600 Physical Configuration

The VAX 8600 processor and console, shown in figure 7-2, are contained in a CPU cabinet and an attached front-end cabinet. The cabinet is 186.0 centimeters (73.25 inches) wide and 153.0 centimeters (60.25 inches) high. The CPU cabinet contains the processor, console logic, floating-point accelerator, space for up to 32 Mbytes of main memory, an SBI adapter bus backplane, and an I/O adapter backplane. The modular power supplies, regulators, and power distribution network are above the logic area. At the bottom of the CPU cabinet are the power control unit, which receives the system line power, and the battery backup unit, which supplies power to the memory in the event of a power failure. A blower unit at the top of the cabinet provides cooling for the power supplies and logic modules. The switches and indicators that monitor and control the system operation are on the panel at the top of the CPU cabinet.

The front-end cabinet includes the RL02, a 10.4-Mbyte removable-media console load device; a UNIBUS backplane mounted in a BA11-A box for communication devices; and a step-down power transformer that is supplied when 50-Hertz main power is used. The step-down transformer has voltage taps to ensure line power compatibility.

A UNIBUS expansion cabinet can be added to expand the system UNIBUS. A synchronous backplane interconnect (SBI) expansion cabinet can be added to extend the SBI interface, which is included in the main cabinet, and two SBI expansion cabinets can be added when a second SBI adapter is used.

The signal cables between the cabinets and the communication and storage devices connect to I/O connector panels located at the rear of the cabinets. The connector panels have plates that can be removed to mount cable connectors.

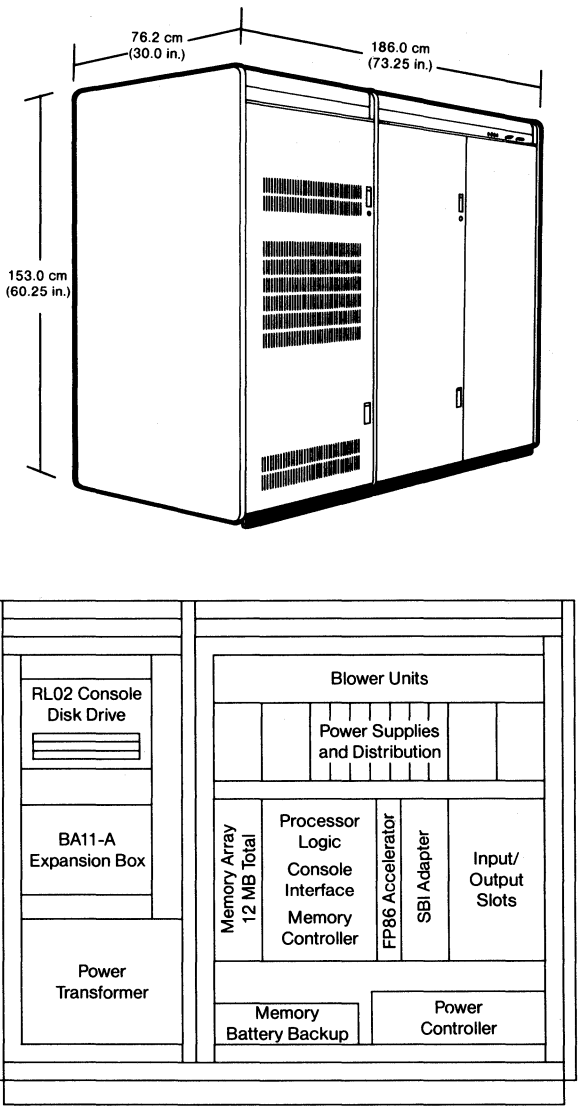


Figure 7-2 ▪ VAX 8600 Processor Configuration

System Expansion Cabinets

The expansion cabinets are available for mounting additional UNIBUS interfaces and for mounting a second SBI adapter.

The H9652-F series of UNIBUS expansion cabinets, shown in figure 7-3, are 153 centimeters (60.25 inches) high and 69.2 centimeters (27.25 inches) wide. The H9652-FA (-FB) cabinet includes a BA11-A mounting box and provides space for adding a second BA11-A mounting box. The H9652-FC (-FD) cabinet includes two BA11-A mounting boxes. No expansion boxes are mounted in the H9652-FE (-FF) cabinet. The BA11-A box contains a power supply and allows installation of DD11-CK (four-slot) and DD11-DK (nine-slot) UNIBUS backplanes. Forty option panel spaces are available at the rear of the cabinets for mounting the I/O device connectors.

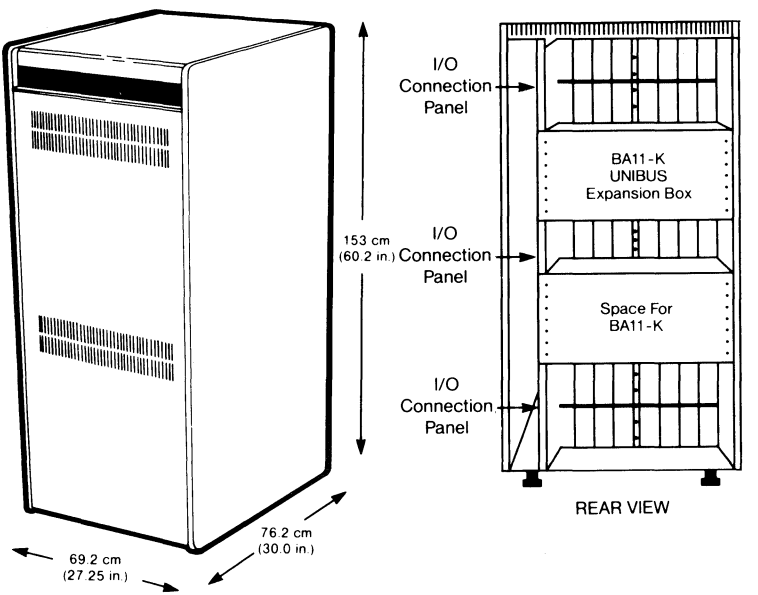


Figure 7-3 ▪ H9652-FA (-FB) UNIBUS Expansion Cabinet Configuration

The H9652-CA (-CB) is a SBI expansion cabinet 153 centimeters (60.25 inches) high and 69.2 centimeters (27.25 inches) wide. The SBI expansion cabinet contains an SBI backplane and power supply. It provides four option panel spaces and space for an additional power supply. The configuration of the H9652-CA (-CB) SBI expansion cabinet is shown in figure 7-4.

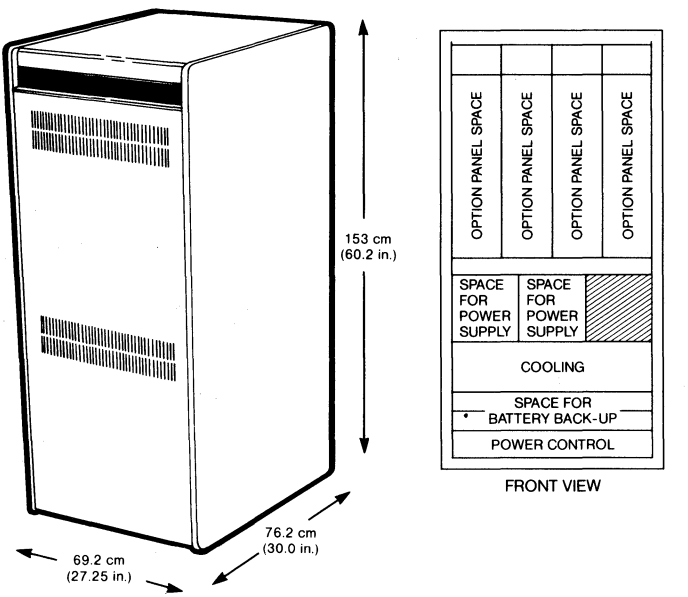


Figure 7-4 ▪ SBI Expansion Cabinet Configuration

VAX 8600 Processor Organization

The functional organization and bus structure of the VAX 8600 processor are shown in figure 7-5. The processor contains four main logic subsystem blocks: the instruction execution logic (Ebox), the instruction/data fetch logic (Ibox), the memory control logic (Mbox), and the floating-point accelerator logic (Fbox). A system of internal buses form the address, data, and control signal paths within the CPU logic. The console subsystem connects to the Ibox through the C bus. The synchronous backplane adapter (SBIA) connects to the Mbox and provides the SBI bus for communication with the external devices.

Console Subsystem

The console subsystem is a microprocessor-controlled interface between the CPU and console terminal, remote diagnostic service port, environmental monitoring module (EMM), and console storage device. It consists of an LA100 console terminal, an RL02 disk drive mounted in the front-end cabinet, and the console interface module located in the CPU backplane assembly. Some of the features and functions of the console subsystem are

- Provides system time-of-year clock
- Performs the system power sequencing and environmental monitoring
- Includes a serial line interface for remote diagnostic service or automatic program testing
- Includes the system and CPU diagnostic programs
- Provides memory storage for bootstrap and diagnostic programs
- Provides self-diagnostic testing during initialization
- Includes powerup configuration programming

The console interface is program-controlled by a T11 microprocessor (PDP-11), and the software in a 256-Kbyte dynamic RAM and a 4-Kbyte PROM. It contains three asynchronous serial-line interfaces, two bus interfaces that communicate with the CPU, and an interface for the RL02 disk drive and the system control panel as shown in figure 7-6.

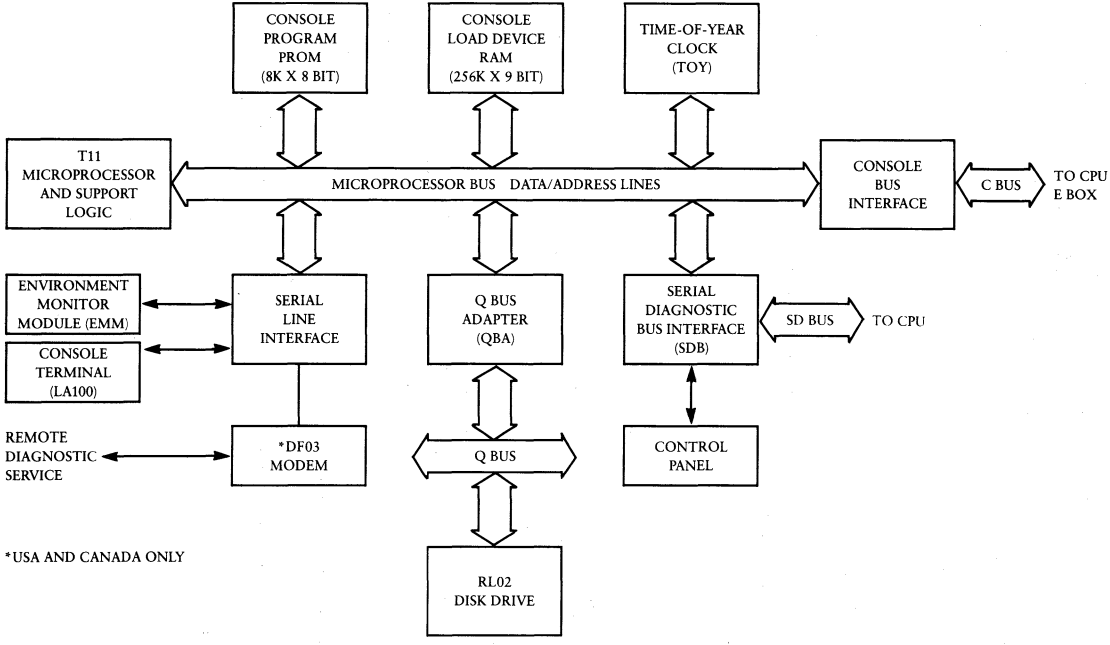


Figure 7-5 ▪ VAX 8600 Processor Components

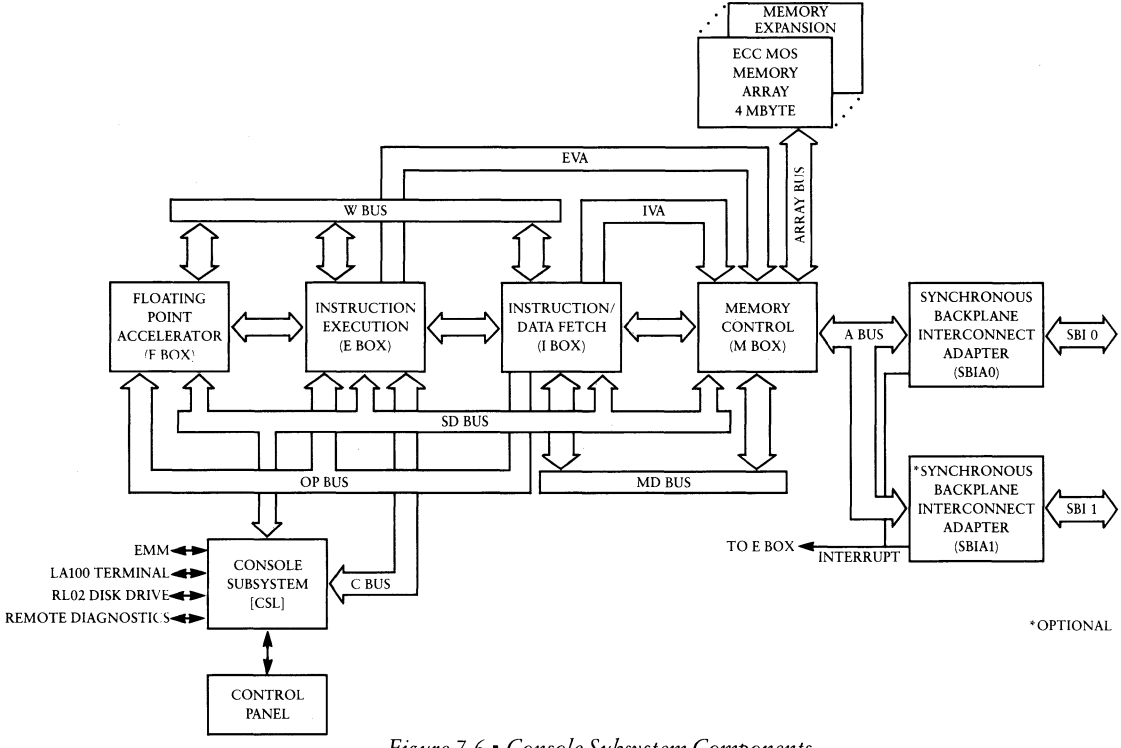


Figure 7-6 ▪ Console Subsystem Components

The console terminal communicates with the console interface through a serial-line interface and serves as an operator's system terminal or as a diagnostic console terminal. As a system terminal, it provides control of the system for initialization, bootstrapping, and updating of the software. The terminal also provides access to the operating system to load system devices and transfer information to EIA serial-line terminals. During the diagnostic function, the terminal can be used to initiate diagnostic programs, to examine and deposit data into the internal processor registers and memory, and to control the processor operation during maintenance.

The remote diagnostic serial-line interface allows diagnostic programs to be loaded and monitored from Digital's remote diagnosis centers. The interface can be connected through a modem to the remote location over standard dialup telephone lines.

The environmental monitoring module (EMM) is located in the power supply area of the CPU cabinet and receives inputs from sensors distributed throughout the system cabinets. The EMM monitors and controls the dc voltages and ground currents within the CPU cabinet, and monitors the environmental conditions, including the cabinet air flow, and cabinet temperatures. The functions of the EMM are controlled by the console interface through the EMM serial-line interface.

The console module contains bus interfaces for communication with the CPU, RL02 disk drive, and system control panel. Information is transferred between the RL02 disk drive and console module through the Q-bus and the Q-bus adapter. The Q-bus connects to the RL02 drive through the RLV12 disk drive controller. The RL02 drive and controller are mounted in the front-end cabinet.

The state of the logic levels in the CPU is monitored by the console interface through the serial diagnostic bus (SDB). Module control bits can also be loaded into the CPU through the SDB. The SDB connects to all major logic elements in the CPU and to the SDB adapter in the console module. It provides 24 visibility/control channels through which the console software can initialize the CPU hardware, increment the CPU clock, sample the state of the CPU logic, and shift the information to the console subsystem for evaluation. Each visibility channel in the CPU contains an 8-bit visibility register that can be shifted serially or loaded in parallel by the channel's clock. The logic levels are selected by the parallel outputs from the visibility register. Parallel data is loaded into a visibility register from the diagnostic terminator network in the CPU module under control of the console subsystem. Serial data is loaded into the visibility register from the console and transferred to the console from the register. The console program reads a visibility channel by stopping the CPU clock and by shifting the

selected bits from the diagnostic terminator network into the visibility register under control of the SDB clock. The progress of the CPU operations can be followed by the continual stepping of the CPU clock and monitoring the results.

The console subsystem also communicates with the CPU through the console bus (C bus). The C bus interface in the console module contains a 32-byte dual-port RAM and three external 8-bit registers. One port in the RAM has direct access to the console software and one port has direct access to the microcode running in the Ebox of the CPU. The RAM locations and the external registers in the C bus interface communicate with the time-of-year clock, the system identity registers, the local and remote terminals, and the RL02 console load device. The interface also contains transmit and receive data buffer locations used by the console-support microcode in the CPU when communicating with the console software.

- ENVIRONMENTAL MONITORING MODULE

The environmental monitoring module (EMM) unit is mounted in a dedicated space of the power supply area and monitors the environment in which the system operates. The EMM unit contains a logic module and a front panel that provides visual indicators. The primary function of the EMM unit is to monitor the dc voltages of the modular power supplies and the outputs of the temperature sensors. The secondary function of the EMM is to measure the flow of air in the CPU cabinet and the current in the grounding lines of the system. Circuits are included to detect when a CPU module is not installed in the backplane or is installed in the wrong location. These conditions are indicated by lights on the front panel of the EMM unit. Indicators are also included to identify a power supply regulator whose thermal switch has been activated.

The EMM contains a 16-channel analog multiplexer, 8-bit digital-to-analog (A/D) converter, analog voltage comparator and a multiprocessor to measure the regulator voltages. Thermistors are used to sense the temperatures, and limit detectors provide outputs that set fault flags when a temperature limit has been exceeded. Voltages and flags are monitored in sequence until a fault is detected. The encoded fault data and an 8-bit A/D value are transferred to the console subsystem. Commands can be issued from the console subsystem to logically disable the regulator output voltage or to set the regulator outputs within specified margins. The system software normally controls the action to be performed when a fault is detected or when a margin is not within the limits. If a thermal switch exceeds 42 degrees Celsius or the logic temperature exceeds 20 degrees Celsius, a warning message will be issued and the system power will be removed if the condition is not corrected within 5 minutes.

Internal Bus Structure—Communication between the logic elements of the CPU is through the internal bus structure consisting of the write bus (W bus), operand bus (OP bus), Ebox virtual-address bus (EVA), Ibox virtual-address bus (IVA), memory data bus (MD bus), and adapter bus (A bus). The Mbox transfers data, address, and select information to memory through the array bus. The console bus (C bus) is a data bus that connects from the console interface to the Ebox. The serial diagnostic bus (SDB) is a serial data path between the console subsystem and the CPU logic elements. Communication between external devices and the CPU is through two SBIA buses that connect to the synchronous backplane interconnect adapters.

The W bus transfers 32 bits of information between the Fbox, Ebox, and Ibox. It is used to update the information in the general purpose registers (GPRs). Whenever a GPR is updated by the Ebox, the resulting data is transferred through the W bus to update the information in the GPR files of the Fbox and the Ibox. The W bus is also used to transfer the data results from the Fbox and Ebox and to store the results in memory through the Ibox. New instruction stream addresses from the Ebox to the Ibox are also sent through the W bus.

The operand bus (OP bus) transfers 33 bits of operand information and CPU register data from the Ibox to the Ebox and Fbox.

Read and write data transfers between main memory and the Ibox, is through the memory data (MD) bus. The MD bus transfers 36 bits of information between the Mbox and the Ibox. The data to the Ebox and Fbox is first transferred through the Ibox.

During read and write operations, the virtual addresses of the data in memory is transferred from the Ibox through the 32-bit IVA for translation by the Mbox. The EVA is used by the Ebox to send virtual addresses to the Mbox for translation during memory read and write operations. It is also used to fill the translation buffer.

During memory read and write operations, the memory address and select information is sent to memory from the Mbox through the memory array bus. The address data is also transferred through this bus.

The C bus is the data path between the data path and the console. It provides the Ebox access to the register file and other control bits in the console. It also passes the ASCII characters between the operators console terminal and the CPU under VMS.

The SDB is used to verify the CPU operation, diagnose and isolate CPU hardware failures, and to transfer information between the console subsystem and the CPU. The SDB consists of 24-bit serial data paths to the CPU logic and each data path contains a module interface, which provides a clock control and data return path to the console. The SDB lines for each channel consist of a clock line, a data line, and three control lines. The control lines consist of a shift line for the visibility channel and two lines used to select the channel.

Processor Logic Elements

The main logic elements in the CPU are the instruction/data fetch (Ibox), the instruction execution (Ebox), the memory control (Mbox), and the floating-point accelerator (Fbox). Each element contains a copy of the 16 general purpose registers (GPRs) to ensure immediate access to the GPR data and system status information.

- INSTRUCTION BOX FUNCTIONS

The instruction box (Ibox), consisting of the instruction and data prefetch logic, is the interface between the Ebox and Fbox execution units and memory. It contains the following logic functions:

- 8-byte FIFO instruction buffer
- Dispatch RAM
- Memory data path
- Control store RAM
- GPR file and an R-log file
- Address and memory data path
- Write latch

All data transfer to and from the execution logic is through the Ibox, which receives the instruction stream of bytes from memory. From the memory information, the Ibox determines the information to retrieve and the activity to initiate in the Ebox. The Ibox performs the first three stages of the four stages required for instruction execution. It prefetches the instruction stream, decodes the instruction opcodes and specifiers, calculates the required address, fetches the operands, and initiates the read and write cycles. The execution of the instruction is then performed by the Ebox or the Fbox while the Ibox begins processing the next instruction.

The instruction buffer is an 8-byte buffer that stores a copy of the bytes fetched from memory. The primary function of the buffer and its associated logic is to prefetch instruction stream bytes and to align them contiguously. More than one instruction is usually contained in the instruction buffer; therefore, the next opcode and specifier will be available when the current instruction is completed. It also is used to fetch and align decimal-string operands.

The dispatch RAM (DRAM) contains a table of dispatch addresses for every macroinstruction and provides control information to the Ibox and address data paths. The address data path and associated control logic perform address calculations and branch displacements, and control the start of the instruction buffer. Using the opcode in the instruction buffer as an address, the DRAM sends a dispatch address to the Ebox control store.

The memory data (MD) path logic controls the data transferred between the Mbox and the Ibox. The data is aligned by a rotator in the data path before transfer to the execution units or to the instruction buffer.

The Ibox control store is a 256 by 52-bit RAM that processes the operand specifiers and branch instructions, updates the register (R-log) file, responds to specific Ebox commands and master reset, and assists in performing the microdiagnostics.

The GPR file contains the same information that is stored in the GPR files of the Mbox and Fbox and provides accessibility to the register data for addressing modes.

The R-log file stores the register information that may change during the instruction execution. Information typically stored in the R-log are current program counter values, the GPR address, the amount of change to a register, and whether the change was added to or subtracted from the register. This information is required during memory management faults to return the processor to the state prior to the condition that caused the fault.

The address data path performs address calculations and branch displacements, and initializes the IBUF. It also provides a backup file for the GPR and macrolevel diagnostics.

A write latch is used to align the data and to store the resulting data from the Ebox and Fbox to be written to cache memory.

- EBOX FUNCTIONS

The Ebox performs logical, arithmetic, and other operations required to execute the instruction. The results are then transferred to the Ibox through the W bus, to be written into memory when required. The Ebox controls the system operation during system initialization and during console mode functions by executing the VAX instruction set and by processing both external and internal interrupts and exceptions. The Ebox includes the following logic elements:

- Data path logic
- Control store RAM
- Microsequencer logic
- Console interface
- Maintenance logic

The data path logic includes a 32-bit binary arithmetic logic unit (ALU), shifter and data packer/unpacker logic, two scratchpad locations that contain copies of the GPRs, a virtual memory queue register, and a W bus register. The ALU performs arithmetic and logical operations and provides special functions to decrease the processing time for division, decimal arithmetic, and compare operations. The shifter and data/packer logic is used to pack and unpack floating-point data, to translate decimal data formats, to shift and rotate arithmetic data, and to perform other bit manipulations. When an external and internal interrupt occur simultaneously, the internal request is processed first. External interrupts, sampled from the I/O adapters and received by the SBIAs, are sent to the Ebox. The Ebox scans the SBIAs to determine the highest priority request pending for each interrupt source. The internal interrupts are received from the console terminal, from the Mbox, and internally from the Ebox. Information is written into the control store RAM when the Ebox clocks are disabled or stalled. The microsequencer logic accesses the RAM in response to console operations and microprogram subroutines, traps, stalls, and subroutines. The console interface provides the path to initialize the control store during the program loading and startup sequence. It also allows the console to enable or disable the maintenance logic and to correct errors.

The control store is an 8K by 92-bit RAM that receives information when the Ebox clocks are stalled or disabled. The microsequencer accesses the control store information in response to microprogram control, instruction dependent dispatching, microprogram subroutines, microtraps, machine stalls, console operations, and microstore-related error conditions. The control store is initialized by the console interface during a cold system start, and allows the console to enable or disable the maintenance logic and perform error correction operations. The maintenance functions are implemented through the clock logic hardware, console software, and the Ebox microword.

The Ebox control consists of context logic, memory port logic, and abort logic. The context logic generates data size information for the Ebox data path and references for the Ebox memory port, and controls W bus transactions. The memory port logic performs functions associated with issuing and requeuing memory commands that are used for branch conditioning and memory management microcode. The Ebox initiates microtraps and creates microtrap vectors from the status information received from the Mbox. Abort logic cancels nonrecoverable operations caused by errors that are detected but do not result in a stall condition. Errors are recovered through the fault process and the processor is returned to the beginning of the microinstruction. The stall logic determines if an input or output stall condition exists in the Ebox microcycle.

The Ebox contains interrupt logic to process internal and external interrupt requests and to determine their priority. The internal interrupts are generated by the console subsystem, the Ebox, and the Mbox. The external interrupts are sampled from the I/O adapters through the A bus.

The Ebox contains two dual-ported scratchpad memories, each consisting of 256 32-bit registers. The scratchpad memories are used as internal temporary storage for the microcode and to store copies of the GPRs and constants. They are also used by the memory management and operation system.

A virtual-memory-queue register provides physical and virtual addresses to the Mbox. The W bus registers are accessed by microcode through the W bus.

The maintenance functions are implemented through the clock logic hardware, console software, and the memory array modules and I/O adapters.

- FLOATING-POINT ACCELERATOR FUNCTIONS

The floating-point accelerator (Fbox) is used to decrease the time to perform floating-point instructions and some integer instructions. The Fbox logic contains the following functions:

- Adder and multiplier logic
- 16 general purpose registers

Operands received from the adder are multiplied by the multiplier logic and the product is returned to the adder for rounding and normalization. During floating-point operations, the multiplier operates on the fractions and the adder operates on the exponent. During integer operations the entire two's complement number is multiplied.

The adder logic performs addition, subtraction, and division and handles the exponents for all the basic operations. The adder contains the logic to unpack the floating-point numbers, to align fractions, and to round and normalize the results. The operands are received from the OP bus and returned to the W bus. The Fbox receives GPR updates and Ebox data, and accesses some Fbox registers through the W bus.

- MEMORY CONTROLLER FUNCTIONS

The memory controller (Mbox) contains the logic for controlling memory operations and for communicating with the I/O subsystems and devices. The main logic elements of the Mbox are

- 16-Kbyte cache memory
- 512-location translation buffer
- Error detection and correction logic
- Port control logic
- Memory arrays
- Mbox control store

The Mbox is the interface to the main memory arrays, to the SBIAs, to the Ibox, and to the Ebox and Fbox. The Mbox connects to the memory through the memory array bus and to the SBIA through the A bus. Its function is to store memory write data and to supply memory read data in response to requests by the Ebox and Ibox, and in response to direct memory access (DMA) requests. It also stores I/O write data and supplies I/O read data in response to requests by the Ebox. During DMA requests from an SBI adapter, the Mbox stores memory write data and supplies memory read data to the SBI. The virtual to physical address translation is performed by the translation buffer. The physical address memory mapping

(PAMM) selects memory array modules and I/O adapters being accessed and is enabled by the PAMM logic in the Mbox. The cache memory provides fast access to the memory data.

The translation buffer is a 512-location cache memory used for storing 256 system-page-table entries and 256 process-page-table entries. During Ebox and Ibox references, the translation buffer is used to decrease the virtual-to-physical translation process. During normal operation the translation buffer is loaded from the page tables in memory by the microcode; during diagnostic operations, loading is through the SD bus.

The PAMM is a 1K by 5-bit RAM that maps the physical address space and is addressed by the high-order bits of the physical address. The output is decoded and used to select the array slot or I/O adapter being accessed. The PAMM is loaded during the system initialization through the SD bus and by system-level programs.

The 16-Kbyte cache is a writeback memory used to decrease the access time of memory references. The cache is arranged in two groups of 8 Kbytes and provides byte parity and error correction code (ECC) on each longword.

The Mbox control store is a 256 by 80-bit RAM that controls the Mbox operations, including port read and write functions, error recovery, cache refill and writeback operations, A bus transactions, register read and write operations, and the diagnostic functions. The control store is loaded from the SD bus during system initialization.

The port control logic provides arbitration during instruction operand fetching of the Ibox and read and write operations by the Ebox. The control functions of the Mbox include special byte-write logic that decreases the time to insert the bytes into longwords. It also includes the memory refresh logic for the 256-Kbyte MOS RAM integrated circuits in the memory array.

Each memory array module contains 4 Mbytes of ECC MOS memory storage and the timing and control logic to refresh memory when the main power to the system fails. During read operations, the physical address from the memory controller is sent to an array together with a start command. The array loads four longwords with ECC from memory into a shift register/bus transceiver. The memory controller then shifts the data, one longword for each cycle, from the register to the A bus. The memory controller performs error correction on each longword it receives. During write operations, the memory controller sends the address and data to the array through the A bus. The data is loaded to the shift register on the memory array, and the memory controller issues a start command to write the data in memory. The memory controller can then initiate a cycle in another array while the data is being written.

- SYNCHRONOUS BACKPLANE INTERCONNECT FUNCTIONS

The synchronous backplane interconnect adapter (SBIA) is the interface between the SBI bus and the A bus. The SBIA provides the functions required to control MASSBUS and UNIBUS I/O operations. It establishes the protocol, provides the timing, and assembles the data for transmission in either direction. The SBIA contains the following logic elements:

- Buffer control and A bus logic
- Register file
- Clock logic
- SBI bus interface
- Interrupt control logic
- Data latches

The SBIA provides the processor and memory nexus functions to the SBI bus and generates the SBI clock signals. The nexus is a physical connection to the SBI bus. Command and address data are exchanged between the SBIA and the Mbox through the A bus. Interrupt information and interrupt polling are exchanged between the SBIA and the Ebox. The A bus and buffer control logic control the reading and writing of the SBIA register files, notify the Mbox when CPU or SBIA transactions occur, receive and buffer the timing signals for the A bus synchronous logic, and notify the Ebox when an interrupt condition is detected.

The register file stores the I/O data transferred through the SBIA. It is a dual-port, 16-line by 32-bit register that operates synchronously with both the A bus and the SBI bus.

The interrupt logic notifies the CPU of errors detected during DMA transactions, fault conditions detected on the SBI bus, and programmed maintenance procedures. It also passes interrupt requests from SBI devices to the CPU.

The clock logic generates the timing signals for the SBI operations.

Data latches are used to hold or convert information transferred to or from the SBI bus. Information from the SBI bus is converted to the A bus format. The SBIA contains 35 registers that are accessed through the CPU transaction buffer. The registers are used to condition the SBIA during bootstrap operations, to record error information, to store vector addresses during the servicing of interrupts and to establish maintenance and diagnostic functions.

The SBI-bus interface logic samples the data transfer lines of the SBI during each cycle and transfers the results to the SBI bus protocol logic and to the data latches. It also transmits information to the SBI bus from the data latches. The SBIA protocol logic provides protocol checking and verifies the commands and data received by the SBIA.

Chapter 8 · VAX Console Subsystems

The console subsystem, which is included with each VAX processor, enables the system user, system manager, and maintenance engineer to effectively communicate with the processor through the console terminal and console command language. The console subsystem is a versatile tool used to initiate the loading of the system operating program and diagnostic programs, to control the system operations, and to perform diagnostic maintenance. It controls the transfer of information to and from the console terminal, the remote diagnostic port, the console disk or tape drive, and the processor control panel. The console subsystems of all VAX processors provide similar basic functions: however, additional capabilities are included with some VAX processor types. Refer to the specific section of this chapter for detailed information related to the console subsystem of each VAX processor. The main features of the console subsystems are as follows:

- Allows the console terminal to functions as an operator's terminal or as a diagnostic maintenance terminal
- Allows communication with the processor through a simple and efficient console command language
- Permits a system to be tested through the remote diagnostic port from a remote Digital facility
- Provides fast and reliable communication with the console disk or tape drive for program loading and updates
- Allows the customer to initiate diagnostics programs
- Enables unattended rebooting of the system program after a power failure

· Console Subsystem Description

The console subsystem is microprocessor-controlled and includes a serial-line interface for the console terminal, remote diagnostic port, and control panel, and an interface for the console load device. The console subsystem can be used to load and update the CPU microcode, to load the operating system programs and diagnostic test programs, and to initiate self-test operations. The console subsystem can be used to halt the processor operation, examine the contents of registers, and load information into any of the registers that are accessible to the user.

The console terminal operates as a user's terminal during normal system operation and as a maintenance device when the processor is not executing instructions. The console terminal is an EIA-compatible ASCII device that enables communication with the processor through the console command language. The console can be a video display or printing terminal and includes a keyboard.

The VAX processor cabinet contains a control panel that operates with the console subsystem. The control panel contains switches and indicators to select the processor operating modes and to indicate the operating status of the system.

The console load device is a disk or magnetic tape drive, depending on the VAX processor type, and is used to load bootstrap programs, processor microcode, and to initiate the loading of the operating system software, software revisions, and diagnostic programs. It also provides a convenient data storage facility for personal system information.

The remote diagnostic port is included with the console subsystem of all VAX processors. The port allows diagnostic evaluation to be performed on the VAX processor system from a remote Digital field service facility. The remote diagnostic service is included with the VAX 8600 system and is optionally available with the other VAX processor systems. The use of this option can significantly decrease the time required to detect malfunctions and to perform routine system maintenance.

In the larger VAX systems, critical environmental conditions are continuously monitored and the information is transferred to the console subsystem for program evaluation and control.

Console Modes of Operation

The console subsystems operate in program I/O mode, in console I/O mode, or in remote diagnostic (RD) mode. One mode is always enabled when the power is applied to the processor.

In program I/O mode, the characters typed on the console terminal are transferred to the processor program and information from the processor is displayed on the console terminal. The console terminal communicates with the operating system program and the CPU interprets instructions, services interrupts and exceptions, and initiates input/output data transfers.

In console I/O mode, the characters typed on the console terminal control the operation of the CPU. Normal CPU operations are suspended and the processor performs operations in response to instructions or commands from the console or remote terminal. Direct memory access (DMA) transactions that are in process between devices and memory when the console mode is initiated will continue to be processed; however, interrupt requests will not be serviced.

RD mode operations are selected by a switch on the control panel of the CPU. When the remote diagnostic service is included, the console subsystem connects to a host computer at a Digital diagnostic center through a modem. Fault detection and preventive maintenance can be initiated by the remote facility. In some VAX processor systems, a microprocessor on the RD module interprets the commands typed from the console terminal or from a terminal at the remote service location.

Console Commands

In console I/O mode, a console command causes a series of events to be performed. When the events are completed, a new command may be entered or the processor may return to program I/O mode. Console commands that are indigenous to specific VAX processors, are defined in the section of this chapter that contains the processor information. The basic console commands enable the following operations to be performed:

- Load the operating system and start the CPU
- Initialize the CPU and begin processing
- Execute a console command a specified number of times
- Examine and deposit information in registers or memory locations
- Restart a halted program
- Halt the CPU operation
- Adjust and control the CPU clock
- Insert a comment in a command
- Read or transfer files from the console loading device
- Deposit data in a console relocation register
- Initiate verification routines
- Invoke an indirect command file
- Specify console command defaults

Console Terminal Prompts

Four types of console prompts can be displayed on the terminal to indicate the mode of operation. The prompt indicates to the operator that the program is waiting for a command. When the RD mode is in effect, the RD indicator is lighted on the control panel of the processor. The prompt characters are listed and defined in table 8-1.

Table 8-1 ▪ Console Prompt Characters

Prompt	Description
$	The CPU is in the run mode and the console subsystem is in the console I/O mode.
⟩⟩⟩	The CPU is in the halt mode and the console subsystem is in the console I/O mode.
ROM⟩	The bootstrap loading sequence for the console subsystem is not able to load the console program.
MIC⟩	The micromonitor program is in control of the console subsystem.

Console Halt Codes

The processor enters console I/O mode as a result of the processor operation or by commands initiated from the console terminal. When console mode is entered, the console terminal displays the address contained in the program counter, a code that identifies the halt condition, and a console prompt symbol (⟩⟩⟩). Table 8-2 defines the halt codes and indicates whether a code is relevant to specific VAX processors.

Table 8-2 • Console Halt Codes

Code	Definition	VAX Processors		
		Small	Medium	Large
00	A *halt* or *single-step* command was entered from the console terminal.		x	x
01	A *test* command was successfully initiated from the console terminal.		x	x
02	*Ctrl-P* was typed on the console terminal.	x	x	x
04	The interrupt stack is not valid or the system control block (SCB) was unable to be read.	x	x	x
05	A CPU double-bus write-error halt has occurred.	x	x	x
06	A *halt* instruction was executed when the processor was in the kernel mode.	x		
06	A *halt* instruction was executed and the processor status longword (PSL) indicated the native mode.		x	x
07	An invalid SCB vector.	x		
07	SCB vector bits 1:0 is equal to 3.		x	x
08	SCB vector bits 1:0 is equal to 2 and the user-control store does not exist.		x	x
0A	A *change mode* (CHMx) instruction was issued from the interrupt stack.	x		
0A	A *change mode* (CHMx) instruction was issued while on the interrupt stack.		x	x
0B	A *change mode* (CHMx) instruction was issued to the interrupt stack.	x		
0B	A *change mode* (CHMx) instruction was issued with the SCB vector bits 1:0 not equal to zero.		x	x
0C	An SCB physical read error was detected.	x		

Code	Definition	VAX Processors		
		Small	Medium	Large
11	During the powerup sequence, the restart parameter block was not located, or the switch that selects the restart is not in the *restart* position.		x	x
12	During the powerup sequence, the warm-start flag is false or the restart switch is not enabled.		x	x
13	During the powerup sequence, 64 Kbytes of valid memory was not located.		x	x
14	During the powerup sequence, a defective boot ROM existed or the boot ROM is not installed.		x	x
15	During the powerup sequence, the cold-start was flag set.		x	x
16	The CPU has halted during the powerup sequence because of the control panel switch position.		x	x
FF	The console microverify test has failed.		x	x

Console Command Language

The console commands are English-language entries or their acceptable abbreviations, entered by the operator at the console terminal. The commands are statements typed on the console terminal when the console subsystem is in console I/O mode. A basic command language is used with all VAX processors. Some commands are restricted to specific processors and some commands perform slightly different operations with each processor. The commands are entered into the system by typing a single character in response to the console prompt. Only one character is necessary to constitute a valid command; however, several commands include qualifiers, addresses, and count values to define the operation of the command. The commands that are restricted to a specific processor are defined in separate sections of this chapter.

Several symbols are used in the command syntax to specify command functions. These symbols enclose numeric values, address locations, data, and other control functions. They are listed in table 8-3. The uppercase functions within the symbols specify the actual typed characters and the lowercase functions specify a selected value. The leading zeros of address, count, and data values can be omitted.

Table 8-3 • Console Command Symbols

Symbol	Function
[ ]	Brackets surrounding part of an expression indicate that part of an expression is optional.
{ }	Braces surrounding part of an expression indicate that one or several of the options listed may be selected.
⟨SP⟩	One typed space (space bar).
⟨count⟩	A 32-bit hexadecimal count.
⟨address⟩	An addressed memory or register location.
⟨data⟩	A numeric data value.
⟨qualifier⟩	A command modifier that is defined for each command. All qualifiers are preceded by a slash (/).
⟨qualifier-list⟩	More than one qualifier may be typed.
⟨CR⟩	Carriage return. A line feed (LF) performs the same function.

The basic command language is listed and defined in table 8-4.

Table 8-4 ▪ Console Command Language

Symbol	Command	Operation
B	*Boot*	Loads the operating or diagnostic software into the system from the specified device.
C	*Continue*	Continues the program execution after a program halt.
D	*Deposit*	Deposits data in a specified register or memory location.
E	*Examine*	Examines data from a specified register or memory location.
F	*Find*	Initiates a search of the main memory for 64 Kbytes of valid memory or for the restart parameter block.
H	*Halt*	Stops the processor from executing macroinstructions after completing the current instruction.
I	*Initialize*	Initializes the processor by setting registers and counters to a specified state.
L	*Load*	Loads data from the specified file into memory.
M	*Microstep*	Executes the specified number of microinstructions.
N	*Next*	Executes the specified number of macroinstructions.
R	*Repeat*	Executes and displays the specified command.
S	*Start*	Starts the execution of a program from a specified address.
SE	*Set*	Sets the console parameter to the specified value.
T	*Test*	Executes a self-test program of the console subsystem.
U	*Unjam*	Initializes the system bus.
X	*Binary load/unload*	Transfers data to or from the specified memory locations (Digital use only).
@	*Indirect*	Reads and executes commands from the specified file.

Control Characters

In console I/O mode, several control characters on the console keyboard are used to control the console operation. These control characters are entered by holding down the *Ctrl* key and typing a character. The control characters and their functions are listed in table 8-5.

Table 8-5 Console Control Characters

Console Character	Function
Ctrl-C, Ctrl-P	Initiates the console I/O mode and halts the processor.
Ctrl-U	Ignores all characters typed after the last carriage return.
Rubout	Deletes the previously typed character.
Ctrl-R	Displays the contents of the input buffer.
Ctrl-O	Terminates the output from the console terminal until the Ctrl-O character is retyped.
Ctrl-S	Stops the console printer.
Ctrl-Q	Restarts the console printer.
ESC	Reexecutes the last command typed on the console terminal.
Return	Terminates the typed line.

Errors and Illegal Characters

Errors and illegal characters typed in the console command string can be corrected using the delete key on the console terminal keyboard. When the delete key is pressed, the console displays a backslash (\) together with the character being deleted and also displays a (\) between the last deletion and the next input character. When a command error is detected, the console responds by typing the following message: ?nn (nn being a code defining the type of error).

System Restart and Reboot Sequence

All VAX console subsystems can automatically restart the processor or can reload the operating system software when power is restored after an interruption or when certain conditions occur in the processor operation. The CPU restart sequence is defined as a "warm-start" and is an attempt to restart the previously loaded operating system. The bootstrap loading sequence is defined as a "cold-start" and is an attempt to reload the operating system software from the system load device. If the restart sequence fails, the bootstrap loading sequence can be automatically initiated.

▪ VAX-11/725 Console Subsystem

The VAX-11/725 console subsystem allows the user to communicate with the processor using the console terminal and console command language. The console subsystem includes a microprocessor-controlled interface that executes a program that controls the console functions, a console terminal, two TU58 tape cartridge drives, and a control panel. The console subsystem also includes the following:

- A remote diagnostic (RD) option that enables diagnostic testing of the system through the remote diagnostic port
- A ROM-resident self-test program that can be invoked at powerup and by console command
- Voltage monitoring circuits that check the dc voltages
- Monitoring circuits to test the UNIBUS AC LO signal
- User initiated console diagnostics test
- Circuits to automatically reboot the operating system when power is restored after a power failure

The console interface contains a microprocessor and ROM that stores the resident part of the console program. It also contains a RAM that stores the console-based microdiagnostic program and the microdiagnostic monitor which is loaded from the TU58 tape drive during the diagnostic operations.

The console terminal can be selected from a series of standard Digital devices, including the VT100 series and VT200 series of video terminals and the LA100 and LA12 printing terminals.

Two TU58 tape cartridge drives are used to load the CPU microcode, boot the system, load files into physical memory, load the diagnostic software, and update the operating system software. They also can be used to store site-specific bootstrap procedures. Both TU58 drives connect directly to the console subsystem. One TU58 drive is located on the front of the processor cabinet and the remaining TU58 drive is located within the cabinet.

The control panel contains switches and indicators used to select operating modes and to display the operating conditions of the processor.

The remote diagnostic port allows the testing of the processor operation from a remote Digital location when the remote diagnostic service has been purchased.

Console Modes

The VAX-11/725 console subsystem operates in program I/O mode or in console I/O mode. In program I/O mode, the console terminal operates the same as other users' terminals connected to the processor, and characters typed on the terminal are transferred to the program. When the CPU is not executing instructions it is halted and the console subsystem is in console I/O mode. Direct memory access (DMA) transactions can occur in the console I/O mode; however, interrupt requests are not serviced. This prevents the loss of data during the DMA transaction when the CPU halts. In program I/O mode, the TU58 drive can be used for data storage and retrieval.

System Control Panel

The VAX-11/725 control panel, located at the front of the CPU cabinet, contains two switches and three indicators lights as shown in figure 8-1. Table 8-6 lists the functions of the switches and indicators.

The *local/remote* switch is a six-position rotary key switch used to control the power to the processor and to select local or remote mode of operation.

The *auto restart* switch is a three-position switch that allows the processor to be manually or automatically restarted during the powerup sequence.

The three indicator lights provide processor operating information, remote service information, and dc power status.

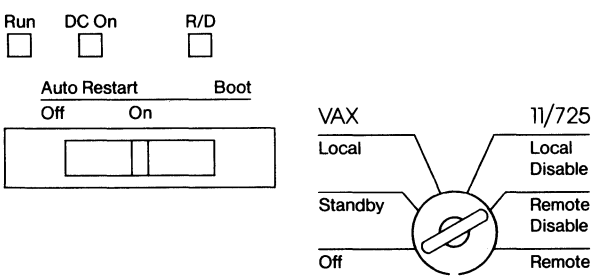


Figure 8-1 • VAX-11/725 Control Panel

Table 8-6 ▪ VAX-11/725 Control Panel Functions

Switch/Indicator	Function
Local/remote switch	*Off*–Removes power from the processor.
	Standby–Disables the processor operation; however, power is applied to maintain the data in the console, writable-control store, main memory, and time-of-year clock.
	Local–In console I/O mode, the console terminal commands control the system operation. In program I/O mode the console terminal operates as a user's terminal and all characters except for *Ctrl-P* are passed to the program. The remote service port is disabled.
	Local disable–In console I/O mode, all characters are ignored. In program I/O mode, all characters are passed to the program. The remote line is disabled.
	Remote disable–In console I/O mode, all characters are ignored. The local terminal is disabled and the remote line is enabled. In program I/O mode, the remote line is enabled so the remote terminal can operate as a user's terminal and all characters are passed to the program.
	Remote–In console I/O mode, the local terminal is disabled when the remote service connection is established. The local terminal can be enabled by commands from the remote service. The console accepts and interprets commands from the remote terminal. In program I/O mode, all characters except *Ctrl-P* are passed to the program from the remote terminal or from the local terminal when it is enabled.

Switch/Indicator	Function
Auto restart switch	*Off*–The console subsystem halts after loading the microcode. The processor remains halted after a halt instruction is executed or power is restored. *On*–The console subsystem loads the microcode and executes a command file that loads the operating system. If a halt condition occurs or if power is restored after a power failure, the console subsystem will attempt to restart the operating system except when a *Ctrl-C* command causes the halt. *Boot* (momentary)–In console I/O mode, the console subsystem executes the boot command file to load the operating system.
Run indicator	Lighted when the CPU is in program I/O mode and is running the main memory program. Not lighted when the main memory program is halted and the CPU is in console I/O mode.
DC on indicator	Lighted when all the dc power supply voltages are within the operating limits. Lighted intermittently when one or more of the dc voltages is not within the operating limits.
R/D indicator	Lighted when the system is connected to the remote diagnostic service. Lighted intermittently during the remote diagnostic service.

Console Control Characters

The control characters used with the VAX-11/725 console subsystem are the same as those listed in table 8-5.

Console Commands

In addition to the console commands available with all VAX processors, the VAX-11/725 includes the *microstep* command and the *directory* command. The *unjam*, *set*, and *indirect* commands are not used. The console commands and command syntax are described as follows:

- *Binary load* command–The *binary load* command causes the console to load binary data into physical memory starting at the specified location. The console processes the command and, if the checksum information is correct, a prompt is issued to the console terminal followed by the binary data requested. A new checksum value is then calculated and processed. This command is specified when bit 31 of the count value is zero. The remaining bits specify the number of bytes to be transferred.

  Syntax: X⟨SP⟩⟨address⟩⟨SP⟩⟨count⟩⟨CR⟩
  ⟨checksum⟩

  –address: the starting address of the load sequence.

  –count: the number of bytes to be transferred.

  –checksum: the two's complement checksum of the command string or binary data

- *Binary unload* command–The *binary unload* command causes the console to unload the binary data from a specified location in physical memory starting at the specified location. This command is specified by a count value with bit 31 set. The remaining bits specify the number of bytes to be transferred.

  Syntax: X⟨SP⟩⟨address⟩⟨SP⟩⟨count⟩⟨CR⟩
  ⟨checksum⟩

  –address: the starting address of unload sequence.

  –count: the binary number of bytes to be transferred plus one byte.

  –checksum: the two's complement checksum of the command string or binary data.

- *Boot* command–The *boot* command causes a boot command file to be read from one of the TU58 console drives, loads the file into a 64-Kbyte block of memory, and executes the bootstrap operation. The *boot command* is used to load the operating system software and diagnostic programs from the mass storage device.

Syntax: BOOT[⟨SP⟩⟨device-select⟩][⟨SP⟩⟨device-name⟩⟨CR⟩

– device-select: DD0 = external TU58 drive. DD1 = internal TU58 drive. If not specified, the device is the TU58 drive previously used to load the microcode.

– device-name: DQn = the unit number of the R80 or RL02 disk drive that contains the system disk. If not specified, the DEFOO.CMD file is read from the specified device and executed.

- *Continue* command–The *continue* command restarts the execution of a halted program at the address currently contained in the program counter if the CPU clock is operating. If it is not operating because of a microcode break in program I/O mode, the clock is started and the console enters program I/O mode.

  Syntax: CONTINUE⟨CR⟩

- *Deposit* command–The *deposit* command enables data to be written into a specified memory or register address. If an address or a data qualifier is not specified, the address and data values from the previous *examine* or *deposit* command are used.

  Syntax: DEPOSIT[⟨qualifier-list⟩]⟨SP⟩⟨address⟩⟨SP⟩⟨data⟩⟨CR⟩

  – qualifier-list (data length): /BYTE = one byte, /WORD = two bytes, /LONG = four bytes.

  – qualifier-list (repetition): /n:⟨count⟩ (the number of locations to be examined plus one).

  – qualifier-list (address): /VIRTUAL = virtual memory, /PHYSICAL = physical memory, /INTERNAL = internal processor registers, /GENERAL = general purpose registers R0 through R11, /CONTROL = writable-control-store (WCS) location, /MACHINE = machine dependent internal registers, /UCODE = console microprocessor memory.

  – address (The symbolic address at which the data will be deposited): nnnnnnnn = a hexadecimal address, PSL = processor status longword, PC = program counter, SP = stack pointer, Rn = a general register, + = the location following the last referenced location in an *examine* or *deposit* command, – = the location preceding the last location referenced in an *examine* or *deposit* command, * = the location last referenced in an *examine* or *deposit* command, (@) = the location addressed by the last location referenced in an *examine* or *deposit* command.

  – data: nnnnnnnn = from one to eight hexadecimal digits to be deposited at the specified address.

- Examine *command*–*The examine* command enables the contents of a specified memory location or register to be examined. If an address is not specified, the address from the previous *examine* or *deposit* command is used.

  Syntax: EXAMINE[⟨qualifier-list⟩]⟨SP⟩⟨address⟩⟨CR⟩

  –qualifier-list: the same as described for the *deposit* command.

  –address: a hexadecimal or symbolic address from which the data will be examined. (Refer to the *deposit* command.)

- *Directory* command–The *directory* command displays the directory of the specified TU58 tape drive.

  Syntax: DIRECTORY⟨SP⟩⟨device-select⟩⟨CR⟩

  –device-select: DD0 = TU58 tape drive 0, DD1 = TU58 tape drive 1.

- *Halt* command–The *halt* command resets the console default conditions after the CPU is halted by the halt codes defined in table 8-2.

  Syntax: HALT⟨CR⟩

- *Initialize* command–The *initialize* command starts the microcode, initializes the TU58 controller and processor, and loads the starting address of the 64-Kbyte block of memory into the stack pointer.

  Syntax: INITIALIZE⟨CR⟩

- *Load* command–The *load* command enables data to be written into a memory location from a specified file. If a qualifier is not specified, the data is loaded into physical memory starting at address zero.

  Syntax: LOAD[⟨qualifier-list⟩]⟨SP⟩⟨filename⟩⟨CR⟩

  –qualifier-list: /S = the starting address of the memory location, /C = writable control store, /P = physical memory.

  –filename: the abbreviation of the filename.

- *Microstep* command–The *microstep* command causes the processor to execute the specified number of microinstructions. If a count is not specified, one microinstruction will be executed.

  Syntax: MICROSTEP[⟨SP⟩⟨count⟩]⟨CR⟩

  –count: a decimal count of the number of instructions to be performed.

- *Next* command–The *next* command causes the processor to execute the specified number of macroinstructions. If a count is not specified, one macroinstruction will be executed.

  Syntax: NEXT[⟨SP⟩⟨count⟩]⟨CR⟩

  –count: a decimal count of the number of instructions to be performed.

- *Repeat* command–The *repeat* command causes the specified command to be repeated until manually stopped by a typing a *Ctrl-C, Ctrl-P,* or a *Break* character on the console terminal.

  Syntax: REPEAT⟨SP⟩⟨command⟩⟨CR⟩

  –command: a *deposit*, *examine*, or *initialize* command.

- *Start* command–The *start* command initiates the execution of programs that do not require the operating system or the diagnostic supervisor. These programs are previously loaded into main memory. If an address is not specified, the current PC address is used.

  Syntax: START[⟨SP⟩⟨address⟩]⟨CR⟩

  –address: the hexadecimal address of the program.

- *Test* command–The *test* command initiates the diagnostic test by locating the ENKAA.EXE program on TU58 drive 0 or drive 1 and by loading the program into memory.

  Syntax: TEST⟨CR⟩

Console Command Errors

Console commands that are entered and cannot be processed will be ignored by the console subsystem and an error code will be displayed on the console terminal. The error code consists of two digits preceded by a question mark (?nn). The condition causing the error must be corrected before the command can be reissued. Table 8-7 lists the console command errors codes and their definitions.

Table 8-7 • Console Command Error Codes

Code	Description
?10	An illegal general-purpose register number.
?11	An illegal access of an internal processor register.
?19	Reference to the next location was preceded by an *examine* or *deposit* command to an internal processor register or to the processor status longword.
?20	An access control violation, translation-not-valid fault, or machine check occurred during a read or write operation.
?30	A binary transfer checksum error.
?33	Unrecognized bootstrap loading device.

Bootstrap Loading Sequence

The bootstrap loading sequence is controlled by the console subsystem and is used to load the operating system into the CPU. The bootstrap loading sequence initializes the CPU and executes a bootstrap command file to load the operating system into processor memory from the specified mass storage device. The sequence is initiated by the following conditions:

- By setting the *auto restart* switch to the *boot* position
- During a powerup sequence when the *auto restart* switch is in the *on* position
- By entering the *boot* command while in console I/O mode
- By executing a halt instruction when the processor is in kernel mode and the *auto restart* switch is in the *on* position
- By executing a move to processor register (MTPR) instruction to the console register that invokes the bootstrap sequence
- When the restart procedure fails and the *auto restart* switch is in the *on* position

During the bootstrap sequence, the console subsystem checks the bootstrap flag, initializes the CPU, loads the address plus a value of 200 of the 64-Kbyte block of valid memory into the stack pointer, determines the CPU options that are installed, and executes the default bootstrap command file (DEFBOO.CMD). The default file loads the operating system from the specified mass storage device. If the bootstrap flag is set during the powerup sequence or when the boot command is executed, the flag will be cleared and then set. If the bootstrap flag is set during the remaining conditions that initiate the bootstrap operation, the CPU normally will halt. The bootstrap sequence is manually initiated by the following procedures.

1. Install the tape cartridge containing the bootstrap program into the TU58 tape drive
2. Apply power to the console terminal
3. Set the *auto restart* switch to the *on* position
4. Set the *local/remote* switch to the *local* or *local disable* position

The terminal then displays the following message:

```
CONSOLE
VERSION XX.XX
```

- SYSTEM RESTART PROCEDURE

When the console subsystem gains control of the VAX-11/725 processor following a power restoration or CPU halt condition, it examines the state of the *auto restart* switch. If the switch is in the *off* position, the console remains in console I/O mode with the processor halted. If the *auto restart* switch is set to the *on* position, the console searches through physical memory, starting at location zero, for a valid restart parameter block (RPB). If the *auto restart* switch is in the *off* position, the console awaits further commands.

A valid RPB consists of four longwords that start on a page boundary. The first longword contains the address of the RPB. The second longword contains the address of the restart routine. The third longword contains a checksum value of the restart code. And the fourth longword contains status information.

If a valid RPB exists, the console examines bit 0 of the fourth longword. When bit 0 is set, it indicates that a restart attempt was previously initiated and failed. The console subsystem then executes the default command file (DEFBOO.CMD) to reboot the system. If bit 0 is clear, the console subsystem sets the bit and loads the RPB address plus a value of 200 into the stack pointer. It also loads general purpose register R12 with a value that indicates the cause of the restart. The restart routine, located at the address specified by the second longword, is then initiated.

▪ VAX-11/730 Console Subsystem

The VAX-11/730 console subsystem performs similar functions and includes the same console hardware as the VAX-11/725 console subsystem.

The VAX-11/730 processor includes two TU58 tape cartridge drives. One drive is located on the front of the processor cabinet and the remaining drive is in the cabinet. Both TU58 tape drives are directly connected to the console subsystem and can be used to perform microlevel and macrolevel diagnostics when some system components are not operating properly.

Console Modes

The operation of the console subsystem of the VAX-11/730 processor in console I/O mode and program I/O mode is the same as the VAX-11/725 console subsystem.

System Control Panel

The VAX-11/730 control panel, shown in figure 8-2, is on the front of the CPU cabinet and contains the two switches and four indicators used to monitor and control the system operation. The functions of the switches and indicators are listed in table 8-8.

The *local/remote* switch is a six-position rotary key switch used to control the power to the unit and to select the local or remote modes of operations.

The *auto restart* switch is a three-position slide switch that controls the automatic restart of the system and the manual bootstrap loading functions.

Three red indicator lights provide information on the processor operation, dc power status, and remote diagnostic service.

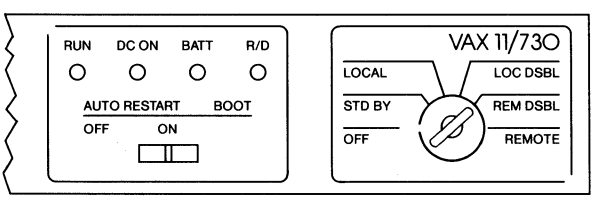


Figure 8-2 ▪ VAX-11/730 Control Panel

Table 8-8 • VAX-11/730 Control Panel

Switch/Indicator	Function
Local/remote switch	*Off*–Removes power from the processor.
	Standby–Disables the processor operation; however, power is applied to maintain the data in the console, writable-control store, main memory, and time-of-year clock.
	Local–In console I/O mode, the console terminal commands control the system operations. In program I/O mode, the console terminal operates as a user's terminal and all characters except for *Ctrl-P* are passed to the program. The remote service port is disabled.
	Local disable–In console I/O mode, all characters are ignored. In program I/O mode, all characters are passed to the program. The remote line is disabled.
	Remote disable–In console I/O mode, all characters are ignored. The local terminal is disabled and the remote line is enabled. In program I/O mode, the remote line is enabled and the remote terminal can operate as user's terminal and all characters are passed to the program.
	Remote–In console I/O mode, the local terminal is disabled when the remote service connection is established. The local terminal can be enabled by commands from the remote service. The console accepts and interprets commands from the remote terminal. In program I/O mode, all characters except *Ctrl-P* are passed to the program from the remote terminal or from the local terminal when it is enabled.
Auto restart switch	*Off*–The console halts after loading the microcode. The processor remains halted after a halt instruction is executed or power is restored.
	On–The console subsystem loads the microcode and executes a command file that loads the operating system. If a halt condition occurs or if power is restored after a power failure, the console subsystem will attempt to restart the operating system except when a *Ctrl-C* command causes the halt.
	Boot (momentary)–In the console I/O mode, the console subsystem executes the boot command file to load the operating system.

Run indicator	Lighted when the CPU is in program I/O mode and is running the main memory program. Not lighted when the main memory program is halted and the CPU is in console I/O mode.
DC on indicator	Lighted when all the dc power supply voltages are within the operating limits. Lighted intermittently when one or more of the dc voltages is not within the operating limits.
R/D indicator	Lighted when the system is connected to the remote diagnostic service. Lighted intermittently during the remote diagnostic service.
Batt indicator	Reserved (no operation).

Console Control Characters
The control characters used with the VAX-11/730 console subsystem are the same as those listed in table 8-5.

Console Commands
The console commands used with the VAX-11/730 console subsystem are the same as those described for the VAX-11/725 console subsystem.

Console Command Errors
Console commands that are entered and cannot be processed will be ignored by the console subsystem and an error code will be displayed on the console terminal. The error code consists of two digits preceded by a question mark (?nn). The condition causing the error must be corrected before the command can be reissued. Table 8-7 lists the console command errors codes and their definitions.

Bootstrap Loading Sequence
The bootstrap loading sequence of the VAX-11/730 processor is initiated by the same conditions and performs the same operations as described for the VAX-11/725 processor.

System Restart Procedures
The system restart procedures for the VAX-11/730 processor are the same as those described for the VAX-11/725 processor.

· VAX-11/750 Console Subsystem

The VAX-11/750 console subsystem includes an LA120 console terminal, a remote diagnostic port, a TU58 cartridge tape drive used as a system bootstrap loading device, and a control panel. The control panel is on the front of the processor unit. The VAX-11/750 console subsystem operates with a subset of the console command language used with the larger VAX systems.

The console interface contains a microprocessor that allows the testing of the system components by console-based microdiagnostics or through a remote testing facility. Some of the diagnostic aids related to the console subsystem are as follows:

- A remote diagnostic (RD) option that allows diagnostic testing of the processor from a remote Digital facility
- ROM-resident self-test that can be invoked at powerup or by console command
- Voltage monitoring circuits that check dc voltages
- Monitoring circuits to check the UNIBUS AC LO signal
- Allows user-initiated diagnostic programs to be performed

The TU58 tape cartridge drive is on the front of the processor cabinet. In console I/O mode, the TU58 drive can be used to load the microcode, to load files in physical memory, and to load diagnostic software and updates to the operating system. It also can be used to store site-specific bootstrap procedures.

Console Modes

The VAX-11/750 console subsystem operates in three separate modes: program I/O mode, console I/O mode, and remote diagnostic (RD) mode. One mode will always be enabled when power is applied to the processor. In program I/O mode, the console terminal operates as a user's terminal and characters typed on the terminal are passed to the processor. In console I/O mode, the console subsystem interprets the commands typed on the console keyboard and performs the functions specified by the commands. In the remote diagnostic mode, the remote terminal controls the processor operations.

System Control Panel

The VAX-11/750 control panel, located on the front of the processor cabinet, contains four switches and seven indicator lights. The switches are used to select the operating modes and to control the console operation, and the lights indicate the state of the processor and the remote diagnostic service. Figure 8-3 shows the location of the switches and indicators and table 8-9 lists their functions.

The *local/remote* switch is a five-position rotary key switch used to control the system power and to select the local and remote modes of operation, depending on the mode of operation of the console subsystem.

The *power on action* is a four-position rotary switch that controls the CPU on a powerup sequence. The switch can be set to restart the system automatically when the power is restored after a power failure.

The *boot device* switch is a four-position rotary switch that is used to select one of four devices from which the system will be bootstrap loaded. The VAX-11/750 CPU can contains up to four ROMs, each containing the code required to bootstrap a device. The switch selects which of the four ROMs that will be used to bootstrap a device.

The *auto restart* switch is a three-position switch that controls the system during the powerup sequence by enabling the automatic bootstrap loading function.

The remote diagnostic indicators are operative only on systems with the remote diagnostic option. The back-lighted words are illuminated during the remote diagnostic procedures.

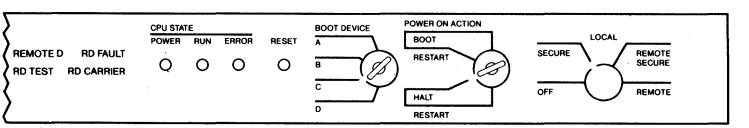


Figure 8-3 ▪ VAX-11/750 Control Panel

Table 8-9 • VAX-11/750 Control Panel Functions

Switch/Indicator	Function
Local/remote switch	*Off*–Removes the power from the CPU. Battery backup power to the memory and the time-of-year clock is present. *Secure*–Disables the console commands. The remote diagnosis unit and the *reset* switch functions are disabled. *Local*–The CPU responds to console commands and the remote diagnosis unit is disabled. *Remote secure*–The console commands are ignored. The remote line replaces the console terminal, and the functions of the *reset* switch are disabled. The remote diagnosis module cannot be used to identify the source of system failures. *Remote*–The console functions are enabled and can be activated only from the remote line; however, the remote line cannot return control to the local terminal.
Power on action switch	*Boot*–Initiates the bootstrap loading sequence from the device selected by the *boot device* switch. *Boot restart*–The system checks for a valid restart parameter block. If a valid block is found, a restart sequence is initiated. If the restart sequence is not successful, a bootstrap sequence is invoked. *Halt*–Halts the processor operation when in console mode. *Halt restart*–The system checks for a valid restart parameter block. If a valid block is found, a restart sequence is initiated. If a valid block is not found or the restart sequence is unsuccessful, the system halts.

Switch/Indicator	Function
Boot device switch	*A*–Initiates the bootstrap loading sequence from the TU58 cartridge tape drive. *B*–Initiates the bootstrap loading sequence from the RK07 disk drive. *C*–Initiates the bootstrap loading sequence from the RL02 disk drive. *D*–Initiates the bootstrap loading sequence from a MASSBUS device.
Reset switch (momentary)	Activates a system powerdown sequence followed by a powerup sequence unless the *local/remote* switch is set to the *secure* position. The system comes up in the state selected by the *power on action* switch. The *reset* switch is used if the system does not respond to console commands.
Power indicator	Lighted green to indicate that the dc power is applied to the CPU.
Run indicator	Lighted green to indicate that the CPU is in program I/O mode.
Error indicator	Lighted red brightly to indicate that the CPU is stopped because of an unrecoverable, control-store parity error. Because console commands are ignored, the *reset* switch must be used to clear the error. Lighted red dimly to indicate that the CPU is functioning normally.
Remote D indicator	Lighted green to indicate that the *local/remote* switch is set to one of the two *remote* positions.
RD Carrier indicator	Lighted green to indicate that the remote diagnostic center has established connection with the console subsystem.
RD Test indicator	Lighted amber to indicate that diagnostic tests are in progress.
RD Fault indicator	Lighted red to indicate that a fault has been detected in the remote diagnosis module.

Console Control Characters

The VAX-11/750 console subsystem responds to the *Ctrl-P*, *Ctrl-U*, and *Ctrl-D* control characters. When the *Ctrl-P* character is typed, the console subsystem enters the console I/O mode, the CPU halts, and a halt message is displayed on the console terminal. Typing the *Ctrl-P* character when the console subsystem is in console I/O mode causes the console default to be reinitialized and displays the halt message on the console terminal. When the *Ctrl-U* character is typed, the console subsystem ignores all characters typed since the previous carriage return. A *Ctrl-D* character typed on the console terminal causes the console subsystem to enter the remote diagnostic mode, if the remote diagnostic module is installed.

Console Commands

The VAX-11/750 console subsystem uses the basic console command set except for the *load*, *find*, *set*, *repeat*, and *unjam* commands. It also includes a *test* command that initiates the microverify routine and initializes the processor.

- *Binary load* command–The *binary load* command causes the console to load binary data into physical memory starting at the specified address. A count value with bit 31 cleared specifies the binary load command, and the remaining bits specify the number of bytes to be transferred. When the console has accepted the command, the number of bytes specified by the count value plus one byte is sent. The last byte is a two's complement checksum of all the data transferred and is calculated by the console. If the checksum information is correct, a prompt is issued to the console terminal, followed by the binary data requested. A new checksum value is calculated and processed.

  Syntax: X⟨SP⟩⟨address⟩⟨SP⟩⟨count⟩⟨CR⟩⟨
  checksum⟩

  – address: the starting hexadecimal address of the load sequence.

  – count: the number of bytes to be transferred plus one byte.

  – checksum: the two's complement checksum of the command string or binary data.

- *Binary unload* command–The *binary unload* command causes the console to unload the binary data from physical memory starting at the specified address. A count value with bit 31 set specifies the binary unload command and the remaining bits specify the number of bytes to be transferred. The console subsystem processes the command and checksum value. If the checksum value is correct, a prompt is issued to the console terminal followed by the number of bytes requested. A second checksum value is then calculated. An error in the checksum value results in an error message sent to the console terminal.

  Syntax: X⟨SP⟩⟨address⟩⟨SP⟩⟨count⟩⟨CR⟩
  ⟨checksum⟩

  –address: the starting hexadecimal address of the unload sequence.

  –count: the binary number of bytes to be transferred.

  –checksum: the two's complement checksum of the command string or binary data.

- *Boot* command–The *boot* command loads the operating system and diagnostic software into memory from the specified mass storage device. The microverify test program normally is performed at the beginning of the bootstrap loading sequence. At the completion of the bootstrap sequence, the console subsystem enters program I/O mode.

  Syntax: BOOT[⟨qualifier-list⟩]⟨SP⟩[⟨device-name,channel,num⟩]⟨CR⟩

  –qualifier-list: /X = inhibit the microverify test, /n = a hexadecimal number that specifies the boot control flags.

  –device-name (a device code that specifies the device from which the program will be loaded): DL = RL02 disk drive, DM = RK06 or RK07 disk drive, DB = RP04, RP05, RP06, RP07, or RM03 disk drives, DD = TU58 tape drive.

  –channel (an I/O channel adapter): A = I/O channel A, B = I/O channel B, C = I/O channel C.

  –num: the number assigned to the device

- *Continue* command–The *continue* command restarts the execution of a halted program at the address currently contained in the program counter if the CPU clock is operating.

  Syntax: CONTINUE⟨CR⟩

- *Deposit* command–The *deposit* command enables data to be written into a specified memory or register address. If an address or data qualifier is not specified, the address and data values from the last *examine* or *deposit* command are used.

  Syntax: DEPOSIT[⟨qualifier-list⟩]⟨SP⟩⟨address⟩⟨SP⟩⟨data⟩⟨CR⟩

  –qualifier-list (data size): /BYTE = one byte, /WORD = two bytes, /LONG = four bytes.

  –qualifier-list (address space): /VIRTUAL = virtual memory, /PHYSICAL = physical memory, /INTERNAL = internal processor registers, /GENERAL = general purpose registers R0 through R15.

  –address (the symbolic address at which the data will be deposited): nnnnnnnn = from one to eight hexadecimal digits, P = processor status longword, * = the address referenced in the previous *deposit* or *examine* command, + = the address following the address referenced in the previous *deposit* or *examine* command.

  –data: nnnnnnnn = from one to eight hexadecimal digits.

- *Examine* command–The *examine* command permits reading of a specified memory location or register. If an address or data-length qualifier is not specified, the physical address and data length value of the previous *deposit* or *examine* command are used.

  Syntax: EXAMINE[⟨qualifier-list⟩]⟨SP⟩⟨address⟩⟨CR⟩

  –qualifier-list: (refer to the *deposit* command).

  –address (the symbolic address from which the data will be examined): (refer to the *deposit* command).

- *Halt* command–The *halt* command resets the console default conditions after the CPU is halted by a halt code (defined in table 8-2).

  Syntax: HALT⟨CR⟩

- *Initialize* command–The *initialize* command initializes the processor to a specified condition, clears and enables the cache memory and translation buffer, clears the program counter, and enters the value of 041F0000 (hexadecimal) into the program counter.

  Syntax: INITIALIZE⟨CR⟩

- *Next* command–The *next* command causes the processor to execute one interrupt stack pointer (ISP) level instruction.

  Syntax: NEXT⟨CR⟩

- *Start* command–The *start* command initiates the execution of programs that do not require the operating system or diagnostic supervisor. The programs start at the address specified, or at the current program counter address if no address is entered.

  Syntax: START⟨SP⟩⟨address⟩⟨CR⟩

  –address: from one to eight hexadecimal digits.

- *Test* command–The *test* command initiates the microverify program and initializes the processor. If an failure is detected, a single-error character is displayed on the console terminal and the processor halts in console I/O mode. An error and halt code is then dispayed on the console terminal.

  Syntax: TEST⟨CR⟩

Console Command Errors

Console commands that are entered and cannot be processed will be ignored by the console subsystem and an error code will be displayed on the console terminal. The error code consists of two digits preceded by a question mark (?nn). The condition causing the error must be corrected before the command can be reissued. Table 8-7 lists the console command errors codes and their definitions. Error codes 10 and 19 are not used in the VAX-11/750 console subsystem and an additional error code 34 is used to indicate an illegal channel adapter.

Bootstrap Loading Sequence

The bootstrap loading sequence is controlled by the console subsystem and is used to load the operating system into the CPU. The bootstrap loading sequence initializes the CPU and executes a bootstrap command file to load the operating system into processor memory from the specified mass storage device. The bootstrap loading sequence is initiated by one the following conditions:

- When power is initially applied or when power is restored after a power failure and the *power on action* switch is in the *boot* position
- By pressing the *reset* pushbutton with the *power on action* switch in the *boot* position
- By entering the *boot* command while in console I/O mode
- By executing a *halt* instruction when the processor is in kernel mode and the *power on action* switch is in the *boot* position
- By executing a move to processor register (MTPR) instruction to the console register that invokes the bootstrap operation
- By a failure of the restart sequence when the *power on action* switch is in the *boot restart* position

During the bootstrap sequence, the console subsystem initializes the CPU, locates a 64-Kbyte block of good memory, and transfers the address of that memory to the device ROM. Block zero of the bootstrap code from the device specified is then transferred to the first page of memory under control of the device ROM. The bootstrap code uses the device ROM as a callable driver for the device and the ROM transfers the files required to start the system operation.

The bootstrap sequence is initiated manually by the following procedures:

1. Apply power to the console terminal.
2. Set the *power on action* switch to *halt* position. The following message will be displayed on the console terminal:

   %%

   00000000 16

   >>>
3. Check that the operating system disk pack is in drive 0 and the *ready* indicator on the drive is lighted.
4. Set the *boot device* switch to the position corresponding to the system device.
5. Set the *power on action* switch to *boot restart* position.
6. Press the *restart* switch

During the bootstrap sequence, the console subsystem performs the following operations. The first two steps of the sequence will not be performed during a powerup condition or when the *restart* pushbutton is pressed and the *power on action* switch is in the *boot* position.

- Clears the cold-start flag and performs the microverify test.
- Initializes the processor status longword.
- Locates a 64-Kbyte block of unused memory.
- Initializes the UNIBUS.
- Loads and validates the first 128 UNIBUS map registers to address the 64-Kbyte block of memory.
- Loads the contents of the boot ROMs into memory.
- Transfers the address of the memory location to the device ROM.
- Checks the cold-start flag. If the flag is set, the operation halts. If the flag is clear, the program sets the flag.
- Loads the input arguments to be used by the device ROM code and operating system.
- Selects the boot ROM indicated by the *boot device* switch.

- SYSTEM RESTART PROCEDURE

When the console subsystem gains control of the VAX-11/750 processor following a power restoration, activation of the reset switch, or CPU halt condition, it examines the state of the *power on action* switch. If the switch is set to either the *halt restart* or *boot restart* position, the console subsystem searches for a valid restart parameter block (RPB) and attempts to restarts the CPU. If the RPB is not located, the console subsystem either halts or performs a bootstrap operation, depending on the position of the *power on action* switch.

A valid RPB consists of four longwords that start on a page boundary. The first longword contains the address of the RPB. The second longword contains the address of the restart routine. The third longword contains a checksum value of the restart code. And the fourth longword contains status information.

If the RPB is located, the console examines bit 0 of the fourth longword. When bit 0 is set, the console subsystem will halt or perform a bootstrap loading sequence, depending on the position of the *power on action* switch. If bit 0 is clear, the console subsystem sets bit 0 and loads the RPB address plus a value of 200 into the stack pointer. It also loads general purpose register R12 with a value that indicates the cause of the restart. The restart routine, located at the address specified by the second longword, is then initiated.

· VAX-11/780 Console Subsystem

The VAX-11/780 console subsystem is controlled by an LSI-11 microprocessor and is the interface between the CPU and the console terminal, remote diagnostic port, control panel, and the RX01 console load device. The console subsystem allows the user to communicate with the processor through the console terminal and console command language. The microprocessor performs diagnostic tests and simplifies the bootstrap operations and system initialization. Some of the features of the console subsytem are as follows:

- The console terminal is a standard ASCII device that can be used as an operator console or user's device.
- The console command language is a powerful, easy-to-use, debugging tool.
- The subsystem allows diagnostics programs to be performed to from a remote Digital service facility.
- The subsystem allows unattended restart of the CPU after a power interruption.

The console subsystem includes an LSI-11 microprocessor, a console interface, a console terminal, a remote diagnostic port, an RX01 floppy disk drive, and a control panel. The subsytem also includes two serial-line interfaces–one for the console terminal and one for the remote diagnostic port.

The LSI-11 microprocessor together with a 4K by 16-bit RAM controls the console subsytem operation. The console interface includes a 4K by 16-bit ROM and controls the transfer of information between the CPU and the devices connected to the console subsystem. It also monitors and controls the operation of the switches and indicators on the control panel.

The console terminal is an LA120 send and receive printing terminal with keyboard. The terminal can be used as a diagnostic terminal or as an operator's terminal during normal system operation. The RX01 is an 8-inch, floppy-disk drive that serves as a console load device for console and diagnostic programs.

The remote diagnostic port enables diagnostic maintenance to be performed on the system from a remote Digital field service location when the remote diagnostic option is include.

Console Modes

The VAX-11/780 console subsystem operates in program I/O mode or console command mode. In program I/O mode, the characters typed on the console terminal are passed to the VAX-11/780 software and information from the software is transferred directly to the console terminal. In console command mode, the console subsystem interprets the console commands and characters and performs diagnostic and maintenance functions.

The VAX-11/780 processor can be either operating or halted during the two console modes. When operating, the processor executes instructions and processes program and device interrupts. When halted by an internal system error or by a halt command from the console terminal, the processor responds to commands issued from the console terminal.

The functions of the console subsystem for each mode are as follows:

- In program I/O mode with the processor operating, characters from the console terminal are transferred through the console subsystem to the processor for interpretation and processor information is transferred directly to the console terminal. The console terminal operates as a user's terminal.
- In console command mode with the processor operating, the microprocessor interprets some console characters and commands from the console terminal to stop or continue the processor operation or to change the console mode.
- In program I/O mode with the processor halted, no system functions can be performed.
- In console I/O mode with the processor halted, the console commands from the console terminal control the system operation. Commands can be used to start and stop the operating system, to modify and display memory and internal register information, to control the processor clock and to initiate diagnostic programs.

System Control Panel

The VAX-11/780 control panel, located on the front of the CPU cabinet, contains three switches and four indicator lights used to monitor and control the processor operation. The functions of the switches and indicators, shown in figure 8-4, are listed in table 8-10.

The *local/remote* switch is a five-position rotary key switch used to control the power and to select the mode of system operation.

The *boot* switch is a momentary switch that initiates the bootstrap program to load the operating system into main memory.

The *auto restart* switch controls the automatic restart of the processor after a power failure or when an internal error condition has halted the processor.

The four indicator lights display the operating conditions of the processor, the status of the power supply, and the status of remote diagnostic service.

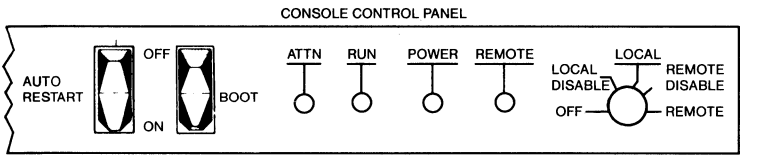


Figure 8-4 • VAX-11/780 Control Panel

Table 8-10 • VAX-11/780 Control Panel Functions

Switch/Indicator	Function
Local/remote switch	*Off*–Removes the power from the CPU. The battery backup power for the time-of-year clock and memory is operating. *Local disable*–Disables console I/O mode and the remote diagnostic access. CPU does not respond to commands from the console terminal. *Local*–Enables the console I/O mode, and the CPU responds to commands from the console terminal. The remote diagnostic access is disabled. *Remote disable*–Enables console I/O mode, and the CPU responds to commands from the console terminal. The remote diagnostic access is enabled. *Remote*–Enables console I/O mode and the remote diagnostic access. The CPU responds to commands from a remote terminal and console commands are ignored.

Auto restart switch	*On*–The CPU is restarted automatically following a power failure or error halt condition. *Off*–Disables the automatic restart of the CPU and the console prompt is displayed on the console terminal following a power failure or error halt condition.
Boot switch (momentary)	*Off*–Disables the console bootstrap sequence. *On*–When pressed, the operating system is bootstrapped and the console enters program I/O mode.
Attn indicator	Lighted red to indicate that CPU is halted and the console subsystem is in console I/O mode.
Run indicator	Lighted green to indicate that the console subsystem is in program I/O mode and the CPU is running.
Power indicator	Lighted green to indicate that the +5 V dc power is applied to the CPU and the *local/remote* switch is not in the *off* position.
Remote indicator	Lighted red to indicate that the CPU is halted and access to the remote diagnostic port is enabled.

Console Control Characters

When the console subsystem is in console I/O mode, special characters typed on the console terminal will control the functions of the console I/O mode. The processor states, console modes, and interaction of the software are shown in figure 8-5.

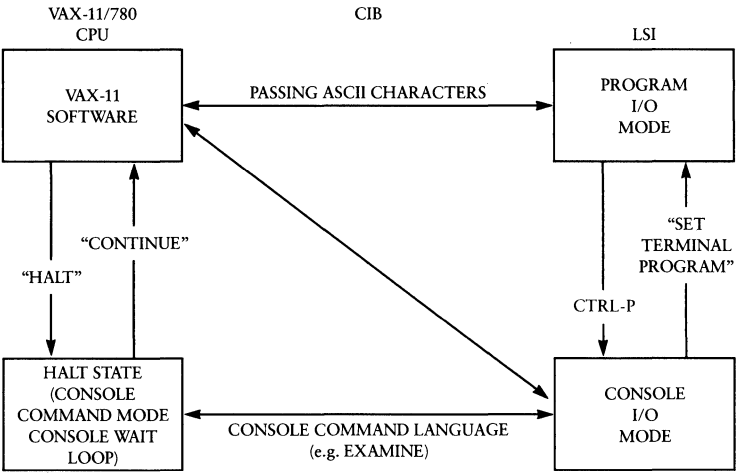


Figure 8-5 • VAX-11/780 Operating Mode Interaction

The console command mode can be entered from the program I/O mode by typing *Ctrl-P* on the console terminal. The program I/O mode can be entered from console command mode by the *set terminal program* command. When the CPU is executing instructions, a subset of console commands is recognized and executed. The functions that the console can perform are limited to those that do not require a direct response from the CPU, except for the *halt* instruction. The console software will not transfer commands to the executing CPU software and will not accept output from the executing CPU software. The console commands that can be used are *clear*, *done*, *examine/V bus*, *halt*, *help*, *set*, *show*, and *wait*. When the CPU is halted and the microprocessor is in console I/O mode, all the functions of the console command set are available and the following operations can be performed:

- Initiate and terminate the execution of the CPU software.
- Display and modify memory elements including main memory, input/output, general purpose register, and process register address space.
- Control the CPU clock to provide single-step clock modes for use in basic hardware or program development.
- Initiate macrodiagnostics and microdiagnostics.

Console Help Files and Commands

Three online help files are included with the console subsystem software and provide a convenient reference for the user. A description of the help files follows:

- A help file, which contains a description of all console commands.
- An abbreviation help file, which contains a list of abbreviations and rules for the console command language.
- An error help file, which contains a list of all error messages for the console command language.

The following commands are used to clear errors, display default conditions, and halt the execution of the command file.

- The *quad clear* command clears uncorrectable error correction codes.
- The *show* command displays the default settings of the data length, address type, radix, and data inputs and outputs. It is also used to display the terminal fill character count and CPU status including the run and halt states and the current clock mode setting.
- The *wait* command is used to halt the execution of the command file until the program running in the CPU is completed. The console then resumes execution of the command file. If the CPU program is halted before completion, the remainder of the command file is not executed.
- Typing the *Ctrl-C* character halts the execution of the remainder of the command file.

Console Commands

The command set used with the VAX-11/780 console subsystem is as follows:

- *Boot* command–The *boot* command initiates the bootstrap sequence to load the operating system program or the diagnostic program into memory from a mass storage device.

  Syntax: BOOT[⟨device-name⟩]⟨CR⟩

  –device-name: DDn = a device designator consisting of two letters and a number assigned to the mass storage device that contains the program to be loaded.

- *Continue* command–The *continue* command restarts the execution of a halted program at the address currently contained in the program counter and enters program I/O mode.

  Syntax: CONTINUE⟨CR⟩

- Deposit command–The *deposit* command enables data to be written into a specified memory location or register. If qualifiers are not specified, the data and address values of the set default command are used.

  Syntax: DEPOSIT[⟨qualifier-list⟩]⟨SP⟩⟨address⟩⟨SP⟩⟨data⟩⟨CR⟩

  –qualifier-list (data length): /BYTE = one byte, /WORD = two bytes, /LONG = four bytes, /QUAD = eight bytes.

  –qualifier-list (address space): /VIRTUAL = virtual memory, /PHYSICAL = physical memory, /INTERNAL = internal processor register, /GENERAL = general purpose registers R0 through PC, / IDBUS = ID bus register, /VBUS = V bus register.

  –address (the symbolic address at which the data will be deposited): PSL = processor status longword, PC = program counter, SP = stack pointer, Rn = general purpose register, + = the location following the last location referenced in a deposit or an examine command, – = the location preceding the last location referenced in a deposit or an examine command, * = the location last referenced in a deposit or an examine command, @ = the address represented by the last location referenced in a deposit or an examine command.

  –data: a hexadecimal value to be deposited at the specified address.

- *Examine* command–The *examine* command enables the contents of a specified memory location or register to be read and displayed on the console terminal. If an address qualifier is not specified, the address from the *set default* command is used.

  Syntax: EXAMINE[⟨qualifier-list⟩]⟨SP⟩[⟨address⟩]⟨CR⟩

  –qualifier-list: the same as the *deposit* command.

  –address (a symbolic address from which the data will be examined): the same as the *deposit* command.

- *Halt* command–The *halt* command stops the operation of the CPU and completes the execution of the instruction in process when the halt command was entered.

  Syntax: HALT⟨CR⟩

- *Indirect* command–The *indirect* command causes the console to access a specified file and to begin executing console commands from the file.

  Syntax: @ ⟨filename⟩⟨CR⟩

  –filename: a unique abbreviation assigned to the file to be accessed.

- *Initialize* command – The *initialize* command initializes the processor to a specified condition, clears the cache memory and translation buffer, and sets the proper values in the program counter and processor status longword.

  Syntax: INITIALIZE⟨CR⟩

- *Help* command – The *help* command accesses and displays a command file that contains a description of all commands and their uses.

  Syntax: HELP⟨CR⟩

- *Abbreviation help* command – The *abbreviation help* command accesses and displays a command file containing a list of abbreviations and rules for the console commands.

  Syntax: ABBREV.HLP⟨CR⟩

- *Error help* command – The *error help* command accesses and displays a file containing a list of error messages for the console commands.

  Syntax: ERROR.HLP⟨CR⟩

- *Load* command – The *load* command is used to read or transfer file data from a console floppy disk to main memory or to the writable-control-store location. If a qualifier is not specified, data is loaded into physical memory at address zero.

  Syntax: LOAD[⟨qualifier-list⟩]⟨SP⟩⟨filename⟩⟨CR⟩

  – qualifier-list: /START:⟨address⟩ = the starting address where the data will be transferred, /WCS = writable control store, /PHYSICAL = physical memory.

  – filename: the abbreviation assigned to the file.

- *Next* command – The *next* command is used to step the CPU clock a number of events. The type of step is specified by the previous *set step* command. If the CPU is in the normal clock mode, it will enter the single-step mode for the duration of the command count.

  Syntax: NEXT[⟨SP⟩⟨count⟩]⟨CR⟩

  – count: the decimal value of the number of clock steps to be performed.

- *Quad clear* command – The *quad clear* command clears an uncorrectable quadword error at the specified physical address.

  Syntax: QCLEAR⟨SP⟩⟨address⟩⟨CR⟩

  – address: the physical memory address.

- *Repeat* command–The *repeat* command causes the specified console command to be repeated until manually stopped by a *Ctrl-C* character typed on the console terminal.

  Syntax: REPEAT⟨command⟩⟨CR⟩

  –command: any console command except the repeat command.

- *Set default* command–The *set default* command establishes the default conditions for the address, data length, and terminal I/O radix for console commands that do not contain a qualifier.

  Syntax: SET⟨SP⟩⟨DEFAULT⟩[⟨SP⟩⟨default-option⟩]⟨CR⟩

  –default-option (address): VIRTUAL = virtual memory, PHYSICAL = physical memory, GENERAL = general purpose registers R1 through R15, IDBUS = console ID bus, VBUS = console V bus.

  –default-option (data): BYTE = one byte, WORD = two bytes, LONG = four bytes, QUAD = eight bytes.

  –default-option (radix): HEX = hexadecimal, OCTAL = octal

- *Set step* command–The *set step* command sets the CPU clock modes as follows:

  Syntax: SET⟨SP⟩STEP[⟨SP⟩⟨step-option⟩]⟨CR⟩

  –step-option: INSTRUCTION = a single-instruction step mode, BUS = a single SBI-cycle step mode, STATE = a single SBI time-state step mode.

- *Set terminal fill* command–The *set terminal fill* command specifies the number of spaces that are transmitted to the console terminal after a ⟨CR⟩ or line feed control.

  Syntax: SET⟨SP⟩⟨TERMINAL FILL⟩⟨count⟩⟨CR⟩

  –count: the hexadecimal number of counts.

- *Set terminal program* command–The *set terminal program* command sets the console terminal to console I/O mode.

  Syntax: SET⟨SP⟩TERMINAL⟨SP⟩⟨PROGRAM⟩⟨CR⟩

- *Set clock* command–The *set clock* command establishes the operating speed of the CPU clock.

  Syntax: SET⟨SP⟩CLOCK[⟨SP⟩⟨speed⟩]⟨CR⟩

  –speed: SLOW = slow speed, NORMAL = normal speed, FAST = fast speed.

- *Set stop on microbreak match* command–The *set stop on microbreak match* command stops the CPU clock when the contents of the microbreak-match register in the console interface is the same as the contents of the CPU micro-pc.

  Syntax: SET⟨SP⟩⟨SOMM⟩⟨CR⟩

- *Set relocation* command–The *set relocation* command deposits data into the console relocation register. The data value is added to the effective address of the virtual and physical memory during the *examine* or the *deposit* commands.

  Syntax: SET⟨SP⟩RELOCATION⟨data⟩⟨CR⟩

  –data: the value of the data to be deposited.

- *Show* command–The *show* command causes the console terminal to display the data inputs and outputs, the default values for the data length, address type, and radix, the terminal fill character count, the CPU status including the run or halt state, and the current clock mode.

  Syntax: SHOW⟨CR⟩

- *Start* commands–The *start* command performs two functions. The first format initializes the CPU, deposits an address in the program counter (PC), and issues a continue instruction to the CPU. The second format deposits an address into the micro-pc and starts the CPU clock in the normal mode of operation.

  Syntax: START[⟨SP⟩⟨address⟩]⟨CR⟩

  –address: the PC address to be deposited.

- Syntax: START/WCS⟨SP⟩⟨address⟩⟨CR⟩

  –address: the micro-pc address.

- *Test* command–The *test* command initates the operation of the microdiagnostic monitor program. Microdiagnostic execution begins immediately if a */COM* qualifier is not entered. If the */COM* qualifier is entered, the microdiagnostic monitor enters its command mode and waits for an operator command. If the microdiagnostic test is successfully completed and no errors are detected, the console I/O mode is reentered. A microdiagnostic floppy must be loaded in the RX01 disk drive.

  Syntax: TEST[⟨/COM⟩]⟨CR⟩

  –/COM: enter the command mode

- *Unjam* command–The *unjam* command clears the fault conditions from the SBI adapter.

  Syntax: UNJAM⟨CR⟩

- *Wait* command–The *wait* command is executed from an indirect command file. The execution of the command file is suspended until one of the following occurs: the program running in the CPU is completed, *@EXIT* is displayed on the console terminal, or the *Ctrl-C* character is typed on the console terminal. The execution of the remainder of the console command file is aborted if the CPU halts and the executing program is not completed.

  Syntax: WAIT⟨SP⟩⟨DONE⟩⟨CR⟩

Console Command Errors

Error messages are displayed on the console terminal when console commands are not valid, when improper command syntax is used, or when error conditions are detected. The console subsystem uses microcode routines in the CPU's control store to perform console functions. Failures in the microroutines are also reported. All console error messages are prefixed by a question mark to distinguish them from informational messages. Table 8-11 lists and defines the error messages. When user interaction is required, the steps appear in parentheses following the respective error description.

Table 8-11 • VAX-11/780 Console Error Messages

Message	Description
Syntactic Errors	
?⟨TEXT-STRING⟩ IS INCOMPLETE	The text string is not a complete console command.
?⟨TEXT-STRING⟩ IS INCORRECT	The text string is not recognized as a valid command.
? FILE NAME ERR	The filename given with a command is invalid.
?IND-COM ERR	The console has detected an error in the format of an indirect command file. The two possible errors are: more than 80 characters in an indirect command line and an indirect command line did not end with a CR or line-feed character.

Command Generated Errors	
Message	**Description**
?FILE NOT FOUND	The filename included with a *load* or @ command does not match a file on the currently loaded floppy disk. This error is also generated when a *help* or *boot* command is issued and the help or boot file cannot be located, and by an attempt to load the writable-control store (WCS) when the WCS file is missing from the floppy disk.
?NO CPU RESPONSE	The specified time interval for the console to wait for a response from the CPU was exceeded. A retry may indicate a possible CPU-related hardware fault.
?CPU NOT IN CONSOLE WAIT LOOP, COMMAND ABORTED	A console command that required assistance from the CPU was issued while the command was not in the console service loop. The CPU must be halted and the command reissued.
?CPU CLOCK STOPPED,COMMAND ABORTED	A console command that requires the CPU clock to be running was issued with the clock stopped. The step mode must be cleared and the command reissued.
CANT DISABLE BOTH FLOPPY'S, FUNCTION ABORTED	An attempt was made to disable both the remote and local floppy.

Microroutine Errors

Message	Description
?MIC-ERR ON FUNCTION	A microerror occurred in the CPU while servicing a console request. The SBI adapter error registers are cleared after this message is printed. The resultant action depends on the error.
?INT-REG ERR	A microerror occurred while attempting to reference a CPU internal processor register. An illegal address will cause this error.
?MICRO-ERROR, CODE = X	An unrecognized microerror occurred. The code returned by the CPU is not in the range of recognized error codes. *X* is the code returned by the CPU.
?MEM-MAN FAULT, CODE = XX	An *examine* or *deposit* command to a virtual memory location caused an error in the memory management microroutine. *XX* is a one-byte error code returned by the routine, with the following bit assignments: Bit 0 = Length violation (bits numbered from right). Bit 1 = Fault was on a page-table-entry reference. Bit 2 = Write or modify intent. Bit 3 = Access violation. Bits 4-7 = Ignored.

CPU Fault-generated Error Messages

Message	Description
?INT-STACK INVALID	The CPU has halted because the interrupt stack is marked invalid.
?CPU DOUBLE-ERR HALT	A machine check has occurred before a previous machine check had been completed. This causes the CPU to execute a double-error halt. The cause of the machine check can be determined by examining ID registers 30-3F (hexadecimal).
?ILL I/E VECTOR	The CPU has detected an illegal interrupt or exception vector.
?NO USR WCS	The CPU has detected an interrupt or exception vector to a nonexistent user's WCS.
?CHM ERR	A change mode instruction has been attempted from the interrupt stack.
INT PENDING	An error was pending when a console-requested halt was performed. A *continue* command must be issued to clear the interrupt.
?MICRO-MACHINE TIME OUT	Indicates that the microprocessor has failed to service an interrupt within a specified time period.

Messages Generated by Floppy Disk Errors

Message	Description
?FLOPPY ERROR,CODE = X	The RX01 driver has detected an error. The codes are printed in hexadecimal radix as follows: X = 1 – Floppy hardware error (CRC, parity, etc.). X = 2 – File not found. X = 3 – Floppy-driver queue overfull. X = 4 – Console software requested an illegal sector number.
?FLOPPY NOT READY	The console floppy drive was not ready when a bootstrap was attempted. The bootstrap operation must be performed again.
?NO BOOT ON FLOPPY	The console has attempted to boot from a floppy disk that does not contain a valid boot block.
?FLOPPY ERROR ON BOOT	A floppy disk error was detected while attempting a console boot.

Version Compatibility Messages

Message	Description
?WARNING-WCS & FPLA VER MISMATCH	The microcode in the WCS is not compatible with the floating-point-accelerator option. This message is printed with each start or continue command from the interrupt-stack pointer.
?FATAL-WCS & PCS VER MISMATCH	The microcode in the programmable-control store is not compatible with the microcode in the writable-control store. The *start* and *continue* commands of the interrupt stack pointer are disabled by the console.
?REMOTE ACCESS NOT SUPPORTED	The console *local/remote* switch is set to the *remote* position and the remote support software is not included in the console.

Console Generated Errors

Message	Description
?TRAP-4, RESTARTING CONSOLE	The console subsystem is restarting after a time-out sequence has occurred.
?UNEXPECTED TRAP MOUNT CONSOLE FLOPPY THEN TYPE ^C	The console is trapped by an unused vector. The console will reboot when *Ctrl-C* is typed on the console terminal.
?Q-BLKD	The output queue of the console output queue is blocked. The console will reboot when the condition is eliminated.

If the *auto restart* switch is in the *on* position and a valid RPB block is located, the console subsystem examines bit 0 of the fourth longword in the RPB. If bit 0 is set, the program halts. If bit 0 is clear, it sets the bit and starts the execution of the restart routine. If a valid memory block is not located, the console subsystem attempts to reboot the system.

- DEFAULT BOOTSTRAP PROCEDURE

The VAX-11/780 contains a default bootstrap command file (DEFBOO.CMD) that is used during normal system operation for the following conditions:

- When a *boot* command that does not specify the device containing the bootstrap program is executed
- During a powerup sequence when the *auto restart* switch is in the *on* position
- During a powerup sequence when the *boot* switch is in the *on* position
- When a move to processor register (MTPR) instruction is issued to the console that invokes the bootstrap program

Bootstrap Loading Sequence

The bootstrap loading sequence is controlled by the console subsystem and is used to load the operating system into the CPU. The bootstrap loading sequence initializes the CPU and executes a bootstrap command file to load the operating system into processor memory from the specified mass storage device.

The bootstrap sequence can be initiated manually by the operator or automatically by the default bootstrap command procedure. For the manual procedure, the operator first invokes the console program by the following procedure:

1. Insert the console bootstrap diskette into diskette drive 0.
2. Set the *auto restart* switch on the control panel to the *off* position.
3. Set the *local/remote* switch on the control panel to *local* position.
4. When power is applied, bootstrap the console diskett, causing display of the console prompt ()))).
5. If the power is already applied, type the *Ctrl–P* character for display of prompt.
6. Enter the *boot* command to reboot the console subsystem.

When the *boot* command is entered, the boot command procedure on the console floppy disk is loaded into the console subsystem and the following operations are performed:

- The CPU is halted and initialized.
- The synchronous backplane interconnect is unjammed.
- The address of the system control block is deposited.
- The general registers are loaded with the information of the device that contains the system program and with the software control flags.

The console subsystem then initiates the program stored in ROM, which results in the following:

- Tests the condition of the CPU and checks for a valid memory configuration.
- Locates a 64-Kbyte block of memory that does not contain uncorrectable errors.
- Loads the address plus a value of 200 (hexadecimal) into the stack-pointer.

The console subsystem then loads the primary bootstap program, transfers control to the program, and enters program I/O mode.

- SYSTEM RESTART PROCEDURE

  After a failure, the console subsystem attempts to restart the CPU when power is restored if the memory information is valid and if the *auto restart* switch is in the *on* position. The restart procedure is also initiated when a *halt* instruction is executed or when a machine error halt condition has occurred and the *auto restart* switch is in the *on* position. The console subsystem loads the console program and microcode from the console floppy.

  If the *auto restart* switch is in the *off* position, the console subsystem issues a console prompt (⟩⟩⟩⟩).

  If the *auto restart* switch is in the *on* position, the console program identifies the type of halt condition and the time of the halt and stores this information in the general purpose registers. It then invokes the restart command file (RESTAR.CMD) on the console floppy disk, which performs the following operations:

  - Deposits the ROM address of the system control block (SCB) in the SCB base processor register when the processor is halted and initialized.
  - Clears the unused general purpose registers (GPR) and deposits the boot device adapter number in GPR (R3). The ROM program then searches memory for a valid restart parameter block.
  - Initiates the ROM program that searches for a valid restart parameter block.

· VAX-11/782 Console Subsystem

The VAX-11/782 system consists of a primary VAX-11/780 processor and an attached, secondary VAX-11/780 processor. Each processor connects to an MA780 shared memory through a memory controller. The local memories of the processors are used only for diagnostic functions. The operation of each processor is controlled by the VAX/VMS operating system. The functions of the console subsystem of the primary and attached processor are similar to those described for the VAX-11/780 pocessor; however, the primary processor controls many of the attached processor operations.

Console Modes

The console subsystem of the primary and attached processor can operate in console I/O mode or program I/O mode. Refer to the console mode descriptions for the VAX-11/780 console subsystem for the detailed information about console I/O mode operation.

System Control Panels

The primary processor and the attached processor each have a control panel on the front of the CPU cabinet. The function of the switches and indicators are the same as those described for the VAX-11/780 processor.

Console Control Characters

The control characters used with the console subsystem of the primary and attached processors are the same as those described for the VAX-11/780 processor.

Console Commands

The console command language, as described for the VAX-11/780 processor, can be used with the primary and attached processor of the VAX-11/782 system. Refer to the user's documentation for detailed information about the commands and their functions.

Console Command Errors

The error messages reported when illegal commands are entered on the console terminal are described in table 8-11.

Bootstrap Loading Sequence

The bootstrap loading sequence for the primary processor is similar to a single VAX-11/780 processor system. The bootstrap loading sequence for the attached processor in the VAX-11/782 system, however, is different from the single VAX-11/780 processor system.

In the VAX-11/782 primary processor, the MA780 shared memory is used during the bootstrap command procedure instead of the local memory. The memory configuration registers are initialized by the command procedure. The MA780 memory controller does not contain a boot ROM to load the system program. The first 64-Kbyte block of shared memory is accessed at physical address zero and the primary bootstrap code file (VMB.EXE) is located at physical address 200.

In the VAX-11/782 attached processor, the bootstrap command procedure initializes the memory registers and then executes a self-branch instruction in the reboot parameter block (RPB) located at physical address zero. After the primary processor is booted, a console command is executed to load the mutiprocessing code. This command also modifies the self-branch instruction in the RPB to point to the attached processor's multiprocessing initialization code.

System Restart Procedure

The restart procedure for the primary processor is similar to the procedure described for the VAX-11/780 single processor except that the boot ROM does not exist in the memory controller and the shared memory is used. The restart command file (RESTAR.CMD) loads the address of the RPB plus a value 200 into the stack pointer.

The restart procedure of the attached processor is the same as that described for the attached processor bootstrap procedure. In the event of a power failure the attached processor returns the current process information to the primary processor.

Default Bootstrap Command Procedure

The VAX-11/782 system uses a modified default bootstrap command file (DFBOO.CMD). A memory ROM is not used to load the stack pointer; and the command file contains a *deposit* command and initialize command for the memory registers.

· VAX-11/785 Console Subsystem

The console subsystem of the VAX-11/785 processor is similar to the VAX-11/780 console subsystem. The VAX-11/785 console subsystem provides improved console performance and includes 48 Kbytes of RAM memory that is used to load the system microcode.

Console Modes

The console subsystem of the VAX-11/785 processor operates in the same modes as the VAX-11/780 processor.

System Control Panel

The control panel for the VAX-11/785 processor is the same as the VAX-11/780 processor. Refer to table 8-10 for the functions of the switches and indicators on the control panel.

Console Control Characters

The control characters used with the console subsystem of the VAX-11/785 processor are the same those for the VAX-11/780 processor.

Console Command Errors

Error messages are displayed on the console terminal when invalid console commands are entered, when improper command syntax is used, and when error conditions are detected. The console subsystem uses microcode routines in the CPU's control store to perform console functions. Failures in the micro routines are also reported. All console error messages are prefixed by a question mark to distinguish them from informational messages. Table 8-11 lists and defines the error messages. When user interaction is required, the steps appear in parentheses following the respective error description.

Bootstrap Loading Sequence

The bootstrap loading sequence can be initiated manually by the operator or automatically by the default bootstrap command. The initialization of the VAX-11/785 system is similar to the VAX-11/780 system except for the differences described in the following paragraphs.

Except for the default bootstrap file, the bootstrap command procedure's file names for the VAX-11/785 are different from those for the VAX-11/780 system. The file names do not contain unit numbers of the devices. When the indirect command file is used to initiate the bootstrap operation, the device unit number must first be entered into the general purpose register R3 before the @ command is issued.

When the *boot* command is used to initiate the bootstrap operation, the console subsystem examines the device-name parameter of *boot* command and performs the action based on the specified parameter. For example, if a device number is not specified or is in the range of 0 through 7 for a *BOOT DU2* command, the device number is ignored and the DUBOO.CMD command file is used. If a device number exceeds 7, the device number must be entered into R3 before the *boot* command is initiated. Device unit numbers that are not specified are considered to be zero.

- DEFAULT BOOTSTRAP COMMAND PROCEDURE

The default bootstrap command procedure file (DEFBOOT.CMD) is contained on the console floppy disk. The bootstrap command file contains a sequence of bootstrap commands used to initialize the system and to load the operating system software into the CPU during normal system operation.

Alternate bootstrap command files can be used in place of the default bootstrap procedure; however, the command files must be edited to deposit the appropriate device unit number in general purpose register R3 before the DEFBOO.CMD file is stored on the console floppy disk. The default procedure is used for the following conditions:

- When a command procedure is not specified during bootstrap operations
- When power is restored after a power failure and the system is in console I/O mode with the *auto restart* switch in the *on* position
- When the *boot* switch is pressed

· VAX 8600 Console Subsystem

The VAX 8600 console subsystem is a programmed interface between the VAX 8600 processor and the console terminal, console disk drive, remote diagnostic port, and the system environmental monitoring module. The console subsystem is controlled by a T-11 microprocessor that supports a subset of the LSI-11 instructions. The console subsystem includes a 256-Kbyte, parity-assisted dynamic RAM, an 8-Kbyte PROM, three programmable serial-line interfaces and a time-of-year clock. The three serial-line interfaces transmit information to and receive information from the console terminal, remote diagnostic line, and environmental monitoring module (EMM).

The console software resides in the 256-Kbyte RAM and the 8-Kbyte PROM. The 8-Kbyte PROM stores the program executed by the T-11 during the powerup sequence. This program self-tests the microprocessor, performs hardware initialization, and loads the console software into the 256-Kbyte RAM from the console load device. The console software includes the diagnostic console program (DCON), macrocode control program (MCP), and the diagnostic control program (DC).

Some of the features and functions of the console subsystem are as follows:

- Includes a system clock control and time-of-year clock with battery backup.
- Performs the system power sequences and monitors the system environment.
- Includes EIA compatible serial-line interfaces for the console terminal, remote diagnostic port, and environment monitoring module.
- Includes the system and CPU diagnostics programs.
- Provides memory storage for bootstrap and diagnostic programs.
- Provides self-diagnostic testing during the system initialization.
- Includes powerup configuration programming

In addition to the standard capabilities provided by all VAX processor subsystems, the VAX 8600 console subsystem provides local diagnostic testing in the diagnostic control mode, a mode to test the microcode, and a debug-trace test facility.

VAX 8600 Console Modes

The console program is loaded from the RL02 disk drive under control of the PROM code during the powerup initialization process. The RL02 is a 10.4-Mbyte cartridge-disk drive mounted in the front-end cabinet that is attached to the CPU cabinet. It consists of an RL02 disk drive and disk drive controller and combines the reliability and convenience of a cartridge disk in a medium-capacity storage device. After the program is loaded and in operation, the console program controls the console subsystem. Both a console diagnostic program and an RL02 diagnostic program are contained on the the console disk.

The console subsystem operates in console I/O mode and program I/O mode. In console I/O mode, the processor is halted and the console subsystem processes the commands issued by an operator from the console terminal. In program I/O mode, the processor is under control of the operating system and the console terminal responds only to requests issued from the processor. Requests from the CPU are serviced by the console subsystem and the console subsystem provides special system monitoring functions. The general features of the console subsystem are as follows:

- Allows independent character transfer between the CPU and the local or remote terminal ports.
- Controls the transfer of information between the CPU and the EMM.
- Controls the logical console requests from the CPU.
- Provides CPU access to the console load device.
- Detects and controls the recovery of CPU error halts, keep-alive failures, and power failures.
- Provides time-of-year information to the CPU.
- Controls the front panel indicators.

Figure 8-6 shows the events required to select the VAX 8600 operating modes. Console I/O mode is entered when the CPU halts and by the following conditions:

- During system powerup and after initialization is completed, if the *boot/halt* switch is in the *halt* position and if the *local/remote* switch is in the *local* position.
- When an attempt to boot or restart the system fails.

- When console subsystem is in program I/O mode and a Ctrl-P character, typed on the console terminal, is allowed by the setting of the control panel switches. If the hexadecimal (HEX) debugger command set of console I/O mode is enabled, the processor will continue to execute instructions and the console program will service the operator requests. If it is not enabled, the Ctrl-P character will halt the processor.

Program I/O mode is entered by the following conditions:

- From the macro context of console I/O mode when a *start* or *continue* command is issued.
- At the completion of the console reboot procedure or after the console program initialization, if an attempt is made for a warm-start or cold-start.

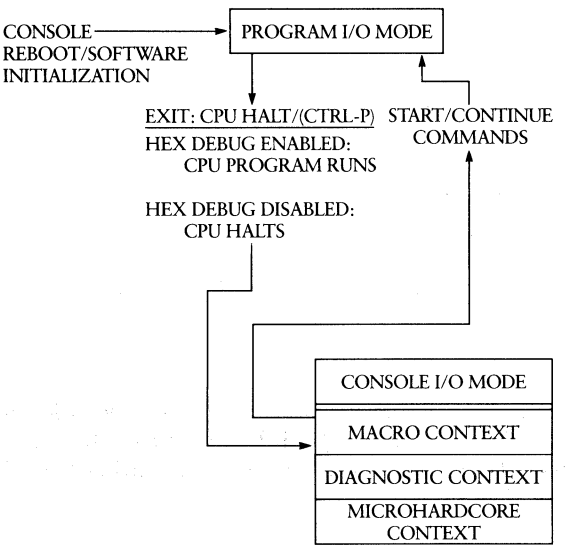


Figure 8-6 • VAX 8600 Operating Mode Selection

The macro context, diagnostic context, or microhardcore context can be selected in console I/O mode. A general command set is available for all console mode contexts and each context has a specific function and command set. A microcode/hardware debugger command set is also available for two of the three contexts. The console subsystem includes local diagnostics in the diagnostics control mode, a mode to test the microcode, and a debug-trace facility. In addition to performing the console terminal I/O operations and processing the commands during console I/O mode, the console subsystem controls the following functions:

- Reporting of unsolicited warning messages from the EMM
- Reporting of the CPU control-store parity errors
- Operation of the front panel switches and indicators

System Control Panel

The VAX 8600 control panel, shown in figure 8-7, is on the front of the CPU cabinet and communicates with the console subsystem through the serial diagnostic (SD) bus. The console subsystem monitors the state of the switches and controls the indicators on the control panel. The position of the switches on the control panel and the existing console mode control the response of the console subsystem to the external inputs. The control panel contains two rotary switches and four status-indicator lights.

The *local/remote* switch is a five-position rotary switch used to control the power to the system and to select access to the console terminal or to the remote terminal by the operating program. When it is in the *local disable* position, the functions of the *boot/halt* switch are disabled.

The *boot/halt* switch is a four-position rotary switch that controls the console program during the console software initialization and in response to a CPU program halt or power failure. When the *local/remote* switch is in the *local disable* position, the *boot/halt* switch is assumed to be in the *restart boot* position. Table 8-12 lists the functions of the switches and indicators on the control panel.

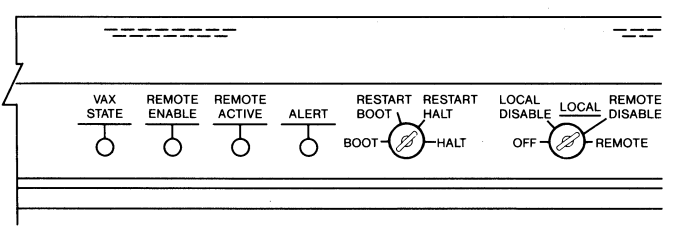


Figure 8-7 ▪ VAX 8600 Control Panel

Table 8-12 • VAX 8600 Control Panel Functions

Switch/Indicator	Function
Local/remote switch	*Off*–The dc power is removed from the CPU and the battery backup unit is disabled. The cooling fans and blowers are not operating. *Local disable*–During the console program initialization, the console program performs an automatic restart bootstrap initialization. In console I/O mode, the console subsystem accepts commands from the console terminal and the remote diagnostic port is disabled. In the program I/O mode, the *Ctrl-P* character typed on the console terminal, is not serviced. *Local*–During console program initialization, the position of the *boot/halt* switch determines the console operation. In console I/O mode, the console subsystem accepts commands from the console terminal and the remote diagnostic port is disabled. In program I/O mode, the *Ctrl-P* character typed on the console terminal will halt the CPU and the console subsystem will enter console I/O mode. *Remote disable*–During console program initialization, the position of the *boot/halt* switch determines the console operation. In console I/O mode, the console subsystem accepts commands from the console terminal and the remote diagnostic port is disabled. In program I/O mode, the console subsystem allows the operation of remote diagnostic service. The *Ctrl-P* character typed on the console terminal will have no effect. *Remote*–During console program initialization, the position of the *boot/halt* switch determines the console operation. In console I/O mode, access to the remote diagnostic port is enabled and the console subsystem accepts commands from both the console terminal and remote diagnostic terminal. In program I/O mode, access to the remote diagnostic port is enabled and the *Ctrl-P* character, typed on the console terminal, will halt the CPU and the console subsystem will enter the console I/O mode.

Boot/halt switch	*Boot* – When the console software initialization is completed and the *local/remote* switch is not in the *local disable* position, the console will attempt to bootstrap the operating system from the default device location specified by the *boot* command. If the bootstrap attempt fails, console I/O mode is entered and the CPU waits for a command from the console terminal. In program I/O mode, if the *local/remote* switch is not in the *local disable* position and the CPU halts, the bootstrap sequence will also be initiated. *Restart boot* – The console subsystem attempts to restart the operating system at the completion of the console software initialization if the *local/remote* switch is not in the *local disable* position. If the restart fails, the bootstrap sequence is initiated. If the bootstrap fails, the console program enters console I/O mode and waits for a command from the console terminal. If the CPU halts in program I/O mode and the *local/remote* switch is not in the *local disable* position, then the bootstrap sequence is initiated. *Restart halt* – The console subsystem attempts to restart the operating system at the completion of the console software initialization if the *local/remote* switch is not in the *local disable* position. If the restart attempt fails, the console program enters console I/O mode and waits for a command from the console terminal. In program I/O mode, if the CPU halts as a result of an error and the *local/remote* switch is not in the *local disable* position, then the console subsystem will enter console I/O mode and wait for a command from the console terminal. *Halt* – At the completion of the console software initialization, the console subsystem enters console I/O mode and waits for a command from the console terminal if the *local/remote* switch is not in the *local disable* position. In program I/O mode, if the CPU halts and the *local/remote* switch is not in the *local disable* position, a program recovery is not attempted.

VAX State indicator	Lighted green intermittently to indicate that the CPU is executing instructions. Not lighted when the CPU is not executing instructions or when the console program is not in program I/O mode.
Remote Enable indicator	Lighted green to indicate the remote diagnostic access is enabled. This condition exists when the *local/remote* switch is in the *remote disable* position and the console program is in program I/O mode.
Remote Active indicator	Lighted green to indicate that the remote terminal is connected and the line is active.
Alert indicator	Lighted red to indicate that the environmental monitoring module has detected a potential failure of an environmental condition in the CPU cabinet. Lighted intermittently to indicate that the internal temperature of the CPU cabinet has exceeded a specified limit or that a problem exists with the cooling air flow. A system failure will result if the condition indicated on the console terminal display is not corrected.

Console Software Programs

The console subsystem performs a complete initialization sequence in response to a console reboot request from the CPU, when a *reboot* command is issued from the console terminal while in console I/O mode and when the ac power is restored after a power interruption. The initialization sequence begins with the execution of the console PROM code and ends with the initialization of the macro context.

The macro context initialization is automatically performed following the console program initialization. It is initially entered by the console program during the powerup sequence, during a console reboot sequence or when the *macro* command, from the general command set, is issued from the console terminal. The macro context is also entered when a program reboot or restart sequence is initiated and the CPU is halted. It provides a superset of the VAX console commands to initialize the processor and to allow the macrocode to operate while the console subsystem is functioning as an interface for the console terminal and EMM inputs.

The diagnostic context provides the commands necessary to support the loading and operation of the microdiagnostic programs. This context, together with the microdiagnostic programs, is used as an aid to processor verification and maintenance after the microhardcore context has been successfully performed.

In the microhardcore context, a group of 98 diagnostic tests can be initiated to check the functions of the VAX 8600 processor hardware. The microhardcore programs are used to verify the operation of the Ebox, Fbox, Ibox, and Mbox logic.

A subset of the general commands set can be used in all contexts to change the existing contexts and to control the basic console functions. The Ctrl-C and Ctrl-P control characters are also used to abort most command lines and all levels of command files. A superset of debug commands can be enabled in console and diagnostic control mode to provides state and clock stepping and monitor, trace, and breakpoint support for hardware and software debugging.

▪ CONSOLE SOFTWARE ORGANIZATION

The console software includes several utility programs used to bootstrap the console software and to initiate and control the diagnostic functions. The console program is loaded from the RL02 disk by the PROM code during the console initialization process. The console software consists of three levels of programs: the diagnostic console (DCON) program, the macro control program (MCP), and the hexadecimal (HEX) debugger program. Figure 8-8 shows the console software organization.

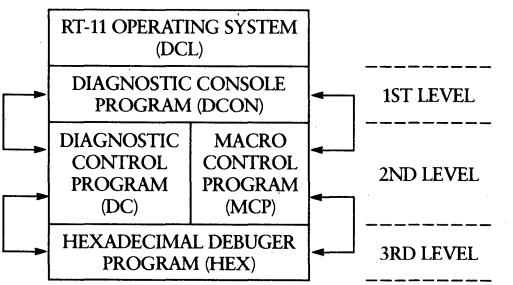


Figure 8-8 ▪ VAX 8600 Console Software Organization

The RT-11 operating system provides the console program with the general system software requirements including a disk file structure, console terminal port services, timer functions, and disk I/O control and bootstrap loading.

The DCON program is an RT-11 based console operating system that is loaded in the T-11 console microprocessor RAM during the system powerup initialization. The program controls the microprocessor and provides a base to load other software programs. It monitors the power and environmental inputs from the system, operates and controls the console terminal and remote terminal, and the RL02 console disk drive, and services the hardware interrupt requests. It also maintains the checksum of the microprocessor memory and provides a runtime library for other software programs.

The MCP is a subset of the DCON program and is part of the software that communicates with the macro ISP machine. The MCP operates with the console support microcode (CSM) in the Ebox to provide the console program with access to various address spaces in the CPU.

The diagnostic control (DC) program is used with the MCP to load, execute, and monitor the results of each microdiagnostic test. The diagnostic support microcode (DSM) is a microprogram in the Ebox and is used with the DC program to link the console to main memory and to the Ebox scratchpad memories.

The hexadecimal (HEX) debugger is a command set that provides predefined visibility registers used to trace the machine states and other functions of the CPU that do not use the console support or diagnostic support microcode.

Console Command Syntax

The syntax of the console subsystem commands of the VAX 8600 processor is similar to the syntax of the other VAX processors. Some of the console commands, however, require additional symbols, qualifiers, switches, and keywords to further define the command. The symbol { } surrounding part of an expression indicates that one selection must be entered from the list of available items. The command syntax can include one or more of the following entries.

- Command – The first item on a command line that defines the command
- /switch – A qualifier used to select an option associated with the command
- /switch:number – A numerical argument associated with the switch
- Keyword – An argument to a command
- Keyword:argument – A numeric or symbolic argument associated with the keyword

General Command Set

The commands in the general command set are available for all console I/O mode contexts and include commands to change the current context, to control the console operation, and to display the contents of files and registers throughout the system. The commands are also used to initialize the EMM modular power system, to reboot the console software, and to reset the CPU.

- *Debug* command–The *debug* command allows the command set of the hexadecimal debugger to be concatenated to the command set of the macro and diagnostic context. This command is not available in the microhardcore context. The CPU clock must be halted before this command can be entered.

  Syntax: DEBUG⟨SP⟩[switch⟩]⟨CR⟩

  –switch: /U start the octal debugger (ODT) program. If ODT is not present, the hex command set is enabled.

- *Diagnose* command–The *diagnose* command changes the console program to the diagnostic context. (Refer to the diagnostic context mode section for the command syntax).

- *Load* command–The *load* command loads the specified file from the system disk to the specified control store or RAM location. If a filename is not specified, the default system microcode file for that control store will be loaded.

  Syntax: LOAD⟨SP⟩{switch}⟨SP⟩[filename[.BPN]]⟨CR⟩

  –switch: /ACCESS, /CONTEXT, /CYCLE, /ECS, /FBACS, /FBMCS, /FDRAM, /ICS, /IDRAM, /MCF, /MCS.

  –filename: the name of the .BPN file to be loaded.

  –default filename: Access = ACCESS.BPN, Context = CTX.BPN, Cycle = CYCLE.BPN, ECS = VENUS.BPN, FBACS = FADD.BPN, FBMCS = FMUL.BPN, FDRAM = FADD.BPN, ICS = IBOX.BPN, IDRAM = VENUS.BPN, MCF = MCF.BPN, MCS = UCODE.BPN.

- *LUPC* command–The *LUPC* command causes the specified microsequencer to be loaded into the specified control-store address.

  Syntax: LUPC⟨SP⟩⟨switch⟩⟨SP⟩⟨hex-addr⟩⟨CR⟩

  –switch: /ECS = Ebox, /ICS = Ibox, MCS = Mbox, /FBACS = Fbox A/FBMCS = Fbox M.

  –hex-addr: the address of the control-store location.

- *Macro* command–The *macro* command changes the console I/O mode to the macro context. Refer to the macro context commands for the command format and functions.

- *MCH* command–The *MCH* (microhardcore) command changes the console I/O mode to the microhardcore context. Refer to the microhardcore context commands for the command format and functions.

- *Reboot* command–The *reboot* command reloads the console software. If the CPU is operating normally when the console program is entered, the console enters program I/O mode. If the CPU is not in normal operation, the function of the command depends on the system state.

  Syntax: REBOOT⟨CR⟩

- *Repeat* command–The *repeat* command allows most console commands to be repeated, except for the commands listed.

  Syntax: REPEAT⟨SP⟩[hex-num]⟨CR⟩

  –hex-num: a hexadecimal number that specifies the number of times the command will be repeated.

  –nonrepeatable commands are *deposit/mark*, *deposit/CSPE*, *repeat*, *next*, *start*, *microstep*, *T micro*, *trace define*, *continue*, *statestep*, *T state*, *trace delete*.

- *Reset* command–The *reset* command initializes the CPU logic by setting the CPU clock and related signals to a specified condition.

  Syntax: RESET⟨CR⟩

- *Restart* command–The *restart* command restarts the console software initialization. When the initialization is completed, the console program enters the macro context.

  Syntax: RESTART⟨CR⟩

- *RT exit* command–The *RT exit* command transfers console control to the RT-11 monitor and then to the console PROM code.

  Syntax: RTEXIT⟨CR⟩

- *Set base* command–The *set base* command causes a base value to be added to the address for all *examine* or *deposit* commands that are issued to virtual memory. If the memory map is disabled, the base value is ignored and the virtual access defaults to a physical memory access.

  Syntax: SET⟨SP⟩BASE[hex-num]⟨CR⟩

  –hex-num: a 32-bit value to be added to the specified address.

- *Set clock frequency* command–The *set clock frequency* command allows the selection of the CPU clock frequency source.

  Syntax: SET⟨SP⟩CLOCK⟨SP⟩FREQ⟨SP⟩{state}⟨CR⟩

  –state: dec-num–a decimal number of the clock frequency from 40 to 64 megahertz, NORMAL–nominal frequency, HIGH–high margined frequency, QUIET–suppresses the reporting of the not locked condition of the variable-control oscillator.

- *Set clock speed* command–The *set clock speed* command selects the operating speed of the clock frequency determined by the *set clock frequency* command.

  Syntax: SET⟨SP⟩CLOCK{speed}⟨CR⟩

  –speed: FULL = the speed set by the *set frequency* command, ONE-FIFTH = one-fifth of the full frequency set by the *set frequency* command, DEFAULT–establishes the last selected frequency and rate as the new defaults for all subsequent INIT/CLOCK commands.

- *Set flag* command–The *set flag* command controls the setting of the software and hardware flags in the console subsystem.

  Syntax: SET⟨SP⟩⟨flag⟩⟨SP⟩{state}⟨CR⟩

  –flag: ABORT provides limited control of the error handling within a command file, ABUS controls the enabling of the A bus, BBU controls the operation of the battery-backup unit, COLD inhibits the repeated unsuccessful attempts to reboot the CPU, EXTI controls the enabling of the external interrupts to the console subsystem, FBOX controls the action of the initialize command to enable or disable the floating-point accelerator, IOSAFE controls the software write-protect feature of console storage device transfers from the CPU, MEMENA controls the state of the console signal that enables the memory bus, SNAP allows the console subsystem to perform a snapshot operation when the the CPU stops executing macro instructions, QUIET provides control of the command-echoing function of the command file processor and control of the output generated by *verify*, *T micro*, and *T state* commands, WARM inhibits repeated unsuccessful attempts to restart the CPU.

  –state: ON, OFF.

- *Set power* command–The *set power* command performs the initialization of the EMM module and the modular power system.

  Syntax: SET⟨SP⟩POWER⟨CR⟩

- *Set somm* command–The *set somm* command controls the enabling and disabling of the micro-mark breakpoint detection in the CPU clock module.

  Syntax: SET⟨SP⟩SOMM⟨SP⟩[switches]⟨SP⟩{state}⟨CR⟩

  –switches: /ECS, /ICS, /MCS, /FIELD.

  –state: ON, OFF.

- *Set terminal* command–The *set terminal* command sets the characteristics of the console terminal and remote diagnostic (RD) terminal ports of the console subsystem.

  Syntax: SET⟨SP⟩TERMINAL⟨SP⟩⟨switch⟩⟨CR⟩

  –switch: /BAUD:nnnn = the transmit and receive baud rate of console terminal port*, /RECEIVE:nnnn = the receive rate of the RD terminal port*, /TRANSMIT:nnnn = the transmit rate of the RD terminal port*, /PASSWORD[Password] = the login password for RD port access, /[NO]DSRS = the data set rate select for low or high modem speed of the RD terminal port, /[NO]PARITY enables or disables the parity generation of the RD terminal port, /[NO]SCOPE = type of terminal device connected to the terminal port; /ODD, /EVEN = parity type for RD terminal port.

  –* nnnn = baud rate 50, 75, 110, 134, 150, 200, 300; 600, 1200, 1800, 2000, 2400, 3600, 4800, 9600, or 19,200.

- *Show clock* command–The *show clock* command displays the current state of the CPU and system clock. It indicates that the CPU and system clocks are operating or disabled, and shows the frequency of the CPU clock source.

  Syntax: SHOW⟨SP⟩CLOCK⟨CR⟩

- *Show file* command–The *show file* command locates the specified file on the RL02 disk drive and displays the content of the file on the console terminal in a binary dump (default) format or in an ASCII format.

  Syntax: SHOW⟨SP⟩[filename]⟨CR⟩

  –filename: [.DAT]⟨SP⟩[/ASCII]

- *Show flags* command–The *show flags* command displays the current state of the console-program control flags. The flags are controlled by the *set flag* command.

  Syntax: SHOW⟨SP⟩FLAGS⟨CR⟩

- *Show panel* command–The *show panel* command displays the setting of the control panel switches and indicators.

  Syntax: SHOW⟨SP⟩PANEL[switch]⟨CR⟩

  –switch: /TEST verifies the read function of the control panel logic.

- *Show power* command–The *show power* command displays the status of the EMM values, the power system, and the system cabinet environmental conditions. This include dc regulator voltages, temperature measurements, air flow sensors, and battery backup unit availability.

  Syntax: SHOW⟨SP⟩POWER⟨CR⟩

- *Show terminal* command–The *show terminal* command displays the state of the console terminal interface and remote diagnostic port of the console subsystem. This includes the state of the EIA signals, scope flag, parity flag, and transmit and receive baud rates of the remote-diagnostic port.

  Syntax: SHOW⟨SP⟩TERMINAL⟨CR⟩

- *Show ucode* command–The *show ucode* command displays the state of the control-store locations and RAMs. The information includes the current .BPN filename, the .BPN filename for the control-store locations, the revision number of the microcode, and the RAM status.

  Syntax: SHOW⟨SP⟩UCODE⟨CR⟩

- *Show version* command–The *show version* command displays the revision information for the CPU hardware, console program, system microcode, console and EMM PROM code, and various software addresses.

  Syntax: SHOW⟨SP⟩VERSION⟨CR⟩

- *Start CPU-clock* command–The *start CPU-clock* command starts the CPU clock. If the system clock is stopped, the command will start the system clock before starting the CPU clock. If the SBIA visibility module is present, the command will also start the SBIA clock.

  Syntax: START⟨SP⟩CPU-CLOCK⟨CR⟩

- *Stop CPU-clock* command–The *stop CPU-clock* command stops the CPU clock but will not affect the operation system clock. If the SBIA visibility module is present, the command will also stop the SBIA clock.

  Syntax: STOP⟨SP⟩CPU-CLOCK⟨CR⟩

- *Start system-clock* command–The *start system-clock* command starts the system clock at the frequency selected by the last *set clock frequency* command.

  Syntax: START⟨SP⟩SYSTEM-CLOCK⟨CR⟩

- *Stop system-clock* command–The *stop system-clock* command stops the system clock and the SBIA clock if the SBIA visibility module is present. If the CPU clock is operating, this command stops the CPU clock before stopping the system clock.

  Syntax: STOP⟨SP⟩SYSTEM-CLOCK⟨CR⟩

- *Unhang* command–The *unhang* command performs an unhang reset sequence that stops and then restarts the CPU clock. It does not clear the error status of the CPU or affect the cache memory.

  Syntax: UNHANG⟨CR⟩

- *Vterm* command–The *Vterm* command provides an aid to isolate faults in the visibility terminator and visibility control logic by configuring the logic to select the specified bit.

  Syntax: VTERM⟨SP⟩[symbol-name, hex-id]⟨CR⟩

  –symbol-name: the V$ symbol assigned to the serial diagnostic bus (SDB) visibility register bit by the SDB CAD (.CDF) files.

  –hex-id: a 16-bit SDB ID code assigned to a visibility signal.

- *Wait* command–The *wait* command prevents the console subsystem from processing commands and from performing other activity for a specified number of 20-millisecond intervals.

  Syntax: WAIT⟨SP⟩[hex-count]⟨CR⟩

  –hex-count: a hexadecimal number between 00 and FF that specifies the number of 20-millisecond intervals.

- *X* command–The *X* command is used to read from and write to the main memory during the automatic communication between the console and other systems. The *X* command protocol is defined by the Digital specification *DEC STD 032*.

  Syntax: X⟨SP⟩⟨hex-addr⟩⟨SP⟩⟨hex-count⟩⟨CR⟩

  –hex-addr: a hexadecimal address of main memory where data is to be transferred.

  –hex-count: the number of bytes to be transferred from the specified address.

- @ command–The @ command causes the command input to be taken from the file indicated by the filename. The outputs generated from the commands in the command file are displayed on the console terminal.

  Syntax: @⟨SP⟩[filename]⟨CR⟩

  –filename: the filename extension is normally .COM; however, other filenames can be specified.

Macro Context Commands

The macro context commands are used when the console subsystem is in macro context mode to initialize the VAX 8600 processor and to allow access to the program I/O mode of processor operation. This context is initially entered by the console program during the powerup sequence and when the macro command is entered from the other console modes. Many of the functions of the macro commands use the console support microcode (CSM).

- *Boot* command–The *boot* command initiates the bootstrap loading of the timesharing code into the VAX 8600 processor from the specified device.

  Syntax: BOOT⟨SP⟩[switches]⟨SP⟩[device]⟨CR⟩

  –switches: /R5:hex-num = a hexadecimal number to be loaded into the R5 register, /NOSTART = do not start the operating system at the completion of the boot command file.

  –device: a one to three character device mnemonic to which is automatically appended the corresponding BOD.COM suffix.

- *Continue* command–The *continue* command performs two separate functions, depending on the state of the CPU. If the CPU is executing macro instructions, it enters program I/O mode. If the CPU is not executing macro instructions, this command causes the instruction execution to begin at the current PC address.

  Syntax: CONTINUE⟨CR⟩

- *Clear memory* command–The *clear memory* command clears the physical memory as specified by the last initialize or physical address memory map (PAMM) command.

  Syntax: CLEAR⟨SP⟩MEMORY⟨CR⟩

- *Deposit* command–The *deposit* command allows data to be deposited into the specified address spaces in the CPU and console subsystem. The data length can be from one to four bytes and the same data can be deposited into successive locations.

  Syntax: DEPOSIT⟨SP⟩[addr]⟨SP⟩[/NEXT:hex-num]⟨SP⟩[data-type]⟨SP⟩{hex-addr, reg-name}⟨hex-data⟩⟨CR⟩

  –addr: /ESCRATCH = Ebox scratchpad RAM, /GENERAL = general purpose register (GPR), /INTERNAL = internal processor registers (IPR), /PAMM = physical address memory map, /PHYSICAL = physical memory, /U = T-11 microprocessor RAM address, /VIRTUAL = virtual memory space.

- hex-num: the next successive location where the same data will be deposited.
- data-type: /BYTE = one byte, /WORD = two bytes, /LONG = four bytes.
- hex-addr (the numeric address of the register to be deposited or one of the following locations): * = the last specified address, + = the address following the last specified address, − = the address preceding the last specified address, @ = the address specified by the contents of the last specified address.
- reg-name: the name of a valid internal processor register, general purpose register, or a miscellaneous register in the processor.
- hex-data: the hexadecimal data to be deposited in the specified address space.

- *Examine* command–The *examine* command allows the contents of several address spaces to be examined. The switch and keywords used in the command syntax are the same as those defined for the *deposit* command, except that ⟨hex-data⟩ information is not used.

  Syntax: EXAMINE⟨SP⟩[addr][/NEXT:hex-num]⟨SP⟩[space]⟨SP⟩[data-type]⟨SP⟩{hex-addr,reg-name}⟨CR⟩

- *Find* command–The *find* command causes a search of the main memory, starting at location zero, for a valid page-aligned 64-Kbyte block of physical memory or for the restart parameter block (RPB). The contents of main memory, accessed during the search operation, are destroyed. The *find/MEMORY* command is used primarily in a boot command file to load the operating systems bootstrap code. The *find/RPB* command initiates a search for the RPB that is used to restart the operating system.

  Syntax: FIND[{switch}]⟨CR⟩

  - switch: /MEMORY = a page-aligned 64-Kbyte block of valid physical memory, /RPB = the restart parameter block.

- *Halt* command–The *halt* command stops the CPU from processing instructions when console I/O mode is entered and the HEX debugger program is in process. If the HEX debugger program is not operating, a halt condition occurs when a *Ctrl-P* character is typed on the console terminal.

  Syntax: HALT⟨CR⟩

- *Initialize* command–The *initialize* command is used to initialize various software and hardware functions of the VAX 8600 processor.

  Syntax: INITIALIZE⟨SP⟩[switch]⟨CR⟩

  –switch: /CLOCK initializes the system clock to operate at 40 megahertz (full speed), /CPU initializes the console support microcode (CSM), /ESCRATCH loads the Ebox scratchpad registers with the values required to start the CPU, /MICRO initializes the microcode, /PAMM configures the PAMM according to the physical memory and the number and type of I/O adapters included with the system, /SDB sets the control channels of the serial diagnostic bus for normal operation.

- *Load main memory* command–The *load main memory* command is used to load the main memory with the binary data from a specified file.

  Syntax: LOAD⟨SP⟩[/START:hex-addr]⟨SP⟩filename[.exe]⟨CR⟩

  –hex-addr: the address where the data will be transferred.

  –filename: the abbreviated name of the file to be accessed.

- *Start* command–The *start* command operates like the *continue* command except that if a ⟨hex-addr⟩ value is specified, it is used as the current program counter (PC) address to start the processor. If the CPU is running when the command is issued, the processor will reenter program I/O mode.

  Syntax: START⟨SP⟩[hex-addr]⟨CR⟩

  –hex-addr: the hexadecimal address at which the processor will start operations.

- *Next* command–The *next* command will single-step the processor the specified number of macroinstructions.

  Syntax: NEXT⟨SP⟩[hex-count]⟨SP⟩

  –hex-count: the hexadecimal number of macroinstructions to be performed.

- *Unjam* command–The *unjam* command clears the unjam register of one of more SBI adapters. If a keyword is not entered, all SBI unjam registers will be cleared.

  Syntax: UNJAM[{adapter}]⟨CR⟩

  –adapter: IOA0 = SBI0 adapter, IOA1 = SBI1 adapter, IOA2 = SBI2 adapter, IOA3 = SBI3 adapter.

- *Verify* command – The *verify* command verifies the contents of any one or all control-store locations by comparing the data read from the location with the contents of the .BPN file. If a switch is not specified, all control-store locations will be verified.

  Syntax: VERIFY⟨SP⟩[switch]⟨CR⟩

  – switch: /ACCESS, /CONTEXT, /CYCLE, /ECS, /FBACS, /FBMCS, /FDRAM, /ICS, /IDRAM, /MCF, /MCS, /PAMM.

Diagnostic Context Commands

The diagnostic context provides commands to support the loading and operation of the microdiagnostics. This context, together with the microdiagnostic programs, is used to verify the processor operation and for maintenance purposes. The diagnostic context is used after the microhardcore context has been successfully performed, to verify the operation of the processor hardcore.

The diagnostic context is entered in response to the *diagnose command.* The diagnostic support microcode (DSM) monitors and controls the microdiagnostic program.

The three control characters used with the diagnostic context are as follows:

- The *Ctrl-P* character interrupts the microdiagnostic program being performed and returns control to the console terminal. Console commands can then be entered to modify the diagnostic environment.
- The *Ctrl-C* character deletes the current console command and returns control to the console terminal.
- The *Ctrl-T* character causes information related to the current diagnostic program to be displayed without interrupting the program operation.

Errors that occur during the performance of the diagnostic context are displayed on the console terminal. The amount of information included in the report depends on the setting of the *remote/local* switch on the control panel and can include the following information:

- Diagnostic name – the test name of the diagnostic program
- First test – the number of the first test performed
- Last test – the number of the last test performed
- Loop goal – the number of passes entered
- Pass count – the number of passes performed
- Ebox scratchpad data – the data and parameters specified by the *set data* command
- Fault detected in test – the number of test that detected the fault

The commands used when operating in the diagnostic context are as follows:

- *Clear data* command–The *clear data* command clears the table of the pointers defined by the *set data* command and is used to define a new set of Ebox scratchpad locations for error reporting.

  Syntax: CLEAR⟨SP⟩DATA⟨CR⟩

- *Continue* command–The *continue* command allows the currently loaded microdiagnostic to resume test execution after being halted by the *set switch* command or by typing the *Ctrl-P* control character.

  Syntax: CONTINUE⟨CR⟩

- *Deposit* command–The *deposit* command allows the modification of the Ebox scratchpad RAM, the system cache, or the W bus RAM.

  Syntax: DEPOSIT⟨SP⟩{switch}⟨SP⟩⟨hex-addr⟩⟨SP⟩⟨hex-data⟩⟨CR⟩

  –switch: /CACHE, /ESCRATCH, /WBUS.

  –hex-addr: the location where the data is to deposited.

  –hex-data: the 32-bit hexadecimal data.

- *Examine* command–The *examine* command allows the contents of the specified Ebox scratchpad locations, cache memory locations, or W bus locations to be examined.

  Syntax: EXAMINE⟨SP⟩{switch}⟨hex-addr⟩⟨CR⟩

  –switch: /CACHE, /ESCRATCH, /WBUS.

  –hex-addr: the hexadecimal address where the data is to be examined.

- *Step* command–The *step* command causes the next test in the currently loaded microdiagnostic program to be immediately executed.

  Syntax: STEP⟨CR⟩

- *Run* command–The *run* command initiates the execution of a command file, provided that the command file does not exceed 80 characters. The switch keywords are described in the *set switch* command.

  Syntax: RUN⟨SP⟩[switch]⟨SP⟩⟨filename⟩⟨CR⟩

  –switch: /BELL: [ON, OFF]; /FAULT: [ISOLATE, LOOP, PAUSE, CONTINUE, IGNORE]; /MODE: [BRIEF, VERBOSE]; /NUMBER: FIRST-TEST, [LAST-TEST]; /PASSES:n (n = number of passes desired)*; /TRACE: [ON, OFF]

  –filename: [.COM].

  * n = the number of times that each test in a diagnostic is performed before the pass is completed.

- *Start* command–The *start* command performs a function similar to that of the *run* command except that the microdiagnostic program must have been previously loaded. The switch selections can be any combination of the keywords listed in the *set switch* command description. The command line is limited to 80 characters.

  Syntax: START⟨SP⟩[switch]⟨CR⟩

  –switch: /BELL: [ON, OFF]; /FAULT: [ISOLATE, LOOP, PAUSE, CONTINUE, IGNORE]; /MODE: [BRIEF, VERBOSE]; /NUMBER: FIRST-TEST, [LAST-TEST]; /PASSES:n (n = the number of passes desired)*; /TRACE: [ON, OFF]

  * n = the number of times each test in a diagnostic is performed before the pass is completed.

- *Set data* command–The *set data* command allows a symbolic name to be assigned to a location in the Ebox scratchpad RAM. This command is normally used in the command files that control the loading of the microdiagnostic code and the setting of the diagnostic control parameters.

  Syntax: SET⟨SP⟩DATA⟨SP⟩⟨escratch-addr⟩⟨SP⟩⟨data-name⟩⟨CR⟩

  –escratch-addr: a hexadecimal address of the Ebox scratchpad RAM from 00 to FF.

  –data-name: a character string of from 1 to 30 characters, to be associated with the Ebox scratchpad data.

- *Set default* command–The *set default* command allows control of the default switch setting that governs the behavior of the diagnostic context and diagnostic support microcode. The switch keywords are described in the *set switch* command.

  Syntax: SET DEFAULT⟨SP⟩[switch]

  –switch: /BELL {ON, OFF}; /FAULT {ISOLATE, LOOP, PAUSE, IGNORE}; /MODE: {BRIEF, VERBOSE}; /NUMBER: FIRST-TEST, [LAST-TEST]; /PASSES:n (n = number of passes desired)*; /TRACE {ON, OFF}

  * n = the number of times each test in a diagnostic is performed before the pass is completed.

- *Set name* command–The *set name* command allows the name of the currently loaded diagnostic to be included as part of a fault report.

  Syntax: SET⟨SP⟩NAME⟨md-filename⟩⟨CR⟩

  –md-filename: a microdiagnostic filename of the command procedure.

- *Set switch* command–The *set switch* command allows control of the default switch settings that determine the functions of the diagnostic control program and the diagnostic support microcode.

  Syntax: SET⟨SP⟩SWITCH⟨SP⟩[switch]⟨CR⟩

  –switch: /BELL [ON, OFF]–controls the audio signal from the console terminal when a fault is detected.

  –switch: /FAULT [ISOLATE]–The tests within the diagnostic are performed as many times as is specified by the PASSES:n switch for the default condition or as specified by the *run* and *start* commands. When an error is detected, an error report is generated and the test continues.

  –switch: /FAULT [LOOP]–The test is performed a specified number of times until a fault is detected and an error report is generated.

  –switch: /FAULT [PAUSE]–The test is halted after the fault is detected and the error report is generated. The test is resumed by issuing a *continue* or a *step* command.

  –switch: /FAULT [CONTINUE]–The tests are performed as many times as is specified by the PASSES:n switch. When an error is detected, an error report is generated and the test continues until all the passes have been completed.

  –switch: /Fault [IGNORE]–When an error is detected, the error report is not generated.

  –switch: /MODE [BRIEF, VERBOSE]–controls the amount of information included in an error report.

  –switch /TRACE [ON, OFF]–controls the enabling (ON) and disabling (OFF) of the test trace facility.

- *Show switches* command–The *show switches* command causes the current switch keyword selections of the *set switch* command to be displayed on the console terminal.

  Syntax: SHOW⟨SP⟩SWITCHES⟨CR⟩

- *Show data* command–The *show data* command causes the symbolic names and associated data to be displayed on the console terminal.

  Syntax: SHOW⟨SP⟩DATA⟨CR⟩

Microhardcore Context Commands

The microhardcore is a diagnostic program used to verify the proper operation of the microhardcore (MCH) programs in the CPU. The MCH program consists of 98 subtests that are grouped into eight CPU-logic function test areas within the CPU. The tests reside on the load media and the MCH program does not create or modify the files in the load device. One test group or all of the tests groups may be selected by the operator after the console diagnostics have been performed successfully. Only the selected tests are loaded into memory. Messages from the MCH program are displayed on the console or remote terminals.

The MCH context is entered by typing *MCH* in reponse to the diagnostic console prompt (>>>). Once the program is initiated, the version name and number will be displayed and the microhardcore prompt ⟨MC⟩ will be issued. A test group or help file, in the format listed as follows, may then be selected.

The microhardcore commands select the microhardcore tests or the help file. Typing a *Ctrl-C* character will stop the diagnostic program at the completion of a subtest. The *START(x)* command tests are performed in one pass and the "⟨MH⟩" prompt is displayed. The *LOOP(x)* command tests are performed repeatedly and are terminated when a *Ctrl-C* character is typed at the end of a subtest.

Syntax: Command⟨SP⟩⟨switch⟩⟨CR⟩

- –Command: DIAG–return to the diagnostic context
- –Command: HELP–display the MHC commands and definitions
- –Command: MACRO–return to the macro context
- –Command: START or LOOP–perform all tests in the normal verify mode
- –Command: STARTQ or LOOPQ–perform all tests in the quick-verify mode
- –Command: STARTC or LOOPC–perform all clock tests
- –Command: STARTE or LOOPE–perform all Ebox tests
- –Command: STARTF or LOOPF–perform all Fbox tests
- –Command: STARTI or LOOPI–performs all Ibox tests
- –Command: STARTL or LOOPL–perform all multibox access tests
- –Command: STARTM or LOOPM–perform all Mbox tests
- –Command: STARTR or LOOPR–perform only the Ebox MCF-CTX RAM tests and the basic microcode tests
- –Command: STARTS or LOOPS–perform only the Ebox SDB and the Ebox E/S tests

– Command: STARTU or LOOPU – perform only the Ebox expanded microcode tests

– switch: BRIEF – display the error report in the short form

– switch: VERBOSE – display the error report in the long form

– switch: QUIET – disables the audio response and permits the error report to be typed on the console terminal

Debug and Trace Facility

The debug and trace facility is used to modify and interrogate the hardware state of the CPU. It can be enabled from the diagnostic context or the macrohardcore context by the *debug* command and includes a hexadecimal debugger (HEX) command set to allow the operator to modify and interrogate the state of the CPU. When it is enabled, an additional bracket (⟩) is added to the normal console command prompt displayed on the console terminal. The commands provided in the HEX command set are as follows:

- *Clear break* command – The *clear break* command clears the specified breakpoint from the appropriate table.

  Syntax: CLEAR⟨SP⟩{table}⟨SP⟩{ident}⟨CR⟩

  – table: ABREAK = the and-break table, OBREAK = the or-break table.

  – ident: V$xxxx = a visibility symbol name, REG = a visibility register name, hex-id = a 16-bit hexadecimal number as defined in the *examine/SDB* command, ALL = removes all breakpoints from the specified table.

- *Clear count* command – The *clear count* command initializes the step-counter to zero. The counter records the number of machine steps (micro or state) since it was last cleared.

  Syntax: CLEAR⟨SP⟩COUNT⟨CR⟩

- *Deposit* command – The *deposit* command is used to modify the contents of control store and RAM locations in the CPU.

  Syntax: DEPOSIT⟨SP⟩[/NEXT:hex-num]⟨SP⟩{switch}⟨SP⟩⟨hex-addr⟩ ⟨hex-data⟩⟨CR⟩

  – /NEXT:hex-num – the number of successive deposits of the same data into consecutive locations.

  – switch: /ACCESS, /CONTEXT, /CYCLE, /ECS, /FBACS, /FBMCS, /FDRAM, /ICS, /IDRAM, /MCF, and /MCS.

  – hex-addr (a hexadecimal number or one of the following symbols): * = the last specified address, + = the address following the last specified address, − = the address preceding last specified address.

– hex-data: a hexadecimal number within the range of the specified control store RAM.

- *Deposit channel* command – The *deposit channel* command deposits the the specified data into the SD bus control channel.

  Syntax: DEPOSIT/CHANNEL⟨SP⟩⟨hex-chnl⟩⟨SP⟩⟨hex-data⟩⟨CR⟩

  – hex-chnl (one of the following channel numbers): 00 = FDA, 09 = EBE, 01 = FBM, 0A = MCC, 03 = IBC, 0D = EBC, 05 = ICA, 0E = CSB, 08 = EDP, 10 = VBA (SBIA visibility module).

  – hex-data: a 16-bit hexadecimal value to be deposited.

- *Deposit/CSPE* command – The *deposit/CSPE* command allows a microword containing a parity error to be deposited into a control-store location. The CPU clock must be stopped before this command can be entered.

  Syntax: DEPOSIT/CSPE⟨SP⟩{switch}⟨SP⟩⟨hex-addr⟩⟨CR⟩

  – switch: /ECS, /FBACS, /FBMCS, /FDRAM, /ICS, /IDRAM and /MCS.

  – hex-addr (a hexadecimal number or one of the following symbols): * = the last specified address, + = the address following the last specified address, – = the address preceding the last specified address.

- *Deposit/mark* command – The *deposit/mark* command allows the control store mark bit to be set or cleared.

  Syntax: DEPOSIT/MARK⟨SP⟩{switch}⟨SP⟩⟨hex-addr⟩⟨SP⟩{state}⟨CR⟩

  – switch: /ECS, /ICS, /MCS.

  – hex-addr (a hexadecimal number or one of the following symbols): * = the last specified address, + = the address following the last specified address, – = the address preceding the last specified address.

  – state: ON, OFF

- *Examine CSAS* command – The *examine CSAS* (control-store-address space) command allows the contents of any control-store RAM location to be examined.

  Syntax: EXAMINE⟨SP⟩[/NEXT:hex-num]⟨SP⟩{switch}⟨SP⟩⟨hex-addr⟩⟨CR⟩

  – /NEXT:hex-num (the hexadecimal number of deposits). Used with other /switch keywords to examine any number of successive locations.

  – switch: /ACCESS, /CONTEXT, /CYCLE, /ECS, /FBACS, /FBMCS, /FDRAM, /ICS, /IDRAM, /MCF, and /MCS.

  – hex-addr (a hexadecimal number or one of the following symbols): * = the last specified address, + = the address following the last specified address, – = the address preceding the last specified address.

- *Examine/channel* command–The *examine/channel* command causes the visibility bits in the specified channel to be displayed on the console terminal.

  Syntax: EXAMINE/CHANNEL⟨SP⟩⟨hex-chnl⟩⟨CR⟩

  –hex-chnl (the SD bus channel number as follows): 00 = FBA,01 = FBM, 02 = MCD, 03 = IBC, 04 = IDP, 05 = ICA, 06 = ICB, 07 = CLK, 08 = EDP, 09 = EBE, 0A = MCC, 0B = MAP, 0C = EBD, 0D = EBC, 0E = CSB, 0F = CSA, 10 = IOA0, 11 = IOA1, 12 = IOA2, 13 = IOA3, 14 = MTM.

- *Examine/SDB* command–The *examine/*SDB (serial diagnostic bus) command causes the name and state of a visibility signal or a visibility register to be displayed on the console terminal.

  Syntax: EXAMINE/SDB⟨SP⟩[ident]⟨CR⟩

  –ident: V$xxxx = a symbol assigned to a visibility signal or register, reg = a visibility register name, hex-id = a 16-bit hexadecimal number of the hardware visibility-bit address or a software defined numerical tag for the visibility register.

- *Exit* command–The *exit command* disables the HEX command set.

  Syntax: EXIT⟨CR⟩

- *Microstep* command–The *microstep* command causes the specified number of microinstructions to be executed at the system clock speed. If the SBI adapter visibility module is present, the SBI clock will be set at one-half the CPU clock speed.

  Syntax: MICROSTEP⟨SP⟩[hex-num]⟨CR⟩

  –hex-num: a hexadecimal step count from 1 to 100.

- *Report* command–The *report* command reads and displays on the console terminal the visibility data in accordance with the current trace list options set by the *trace* command.

  Syntax: REPORT⟨CR⟩

- *Set break* command–The *set break* command sets a break condition in the and-break table or the or-break table. Up to four breakpoints can be set in each table.

  Syntax: SET⟨SP⟩{table}⟨SP⟩{ident}[param]⟨CR⟩

  –table: ABREAK = and-break table, OBREAK = or-break table,

– ident: V$xxxx = a symbol assigned to a visibility signal or register, reg = a visibility register name defined within HEX debugger or specified by the *trace define* command, hex-id = a 16-bit hexadecimal number of the hardware visibility-bit address or a software-defined numerical tag for the visibility register.

– param: SPAN causes a break condition to occur when the signal or register state is different from the state at the start of the trace, VALUE:val causes a break condition to occur when the specified signal or register equals the value specified.

- *Set margin* command – The *set margin* command enables the voltage output levels of the power supply to be varied. This command is used during maintenance functions to detect hardware fault conditions.

  Syntax: SET⟨SP⟩MARGIN⟨SP⟩{value}⟨SP⟩[{reg-mar}]⟨CR⟩

  – value: HIGH = 5 percent above normal voltage, NORMAL = normal voltage, LOW = 5 percent below normal voltage.

  – reg-mar (selects the power-supply regulator outputs as follows): A = regulator A, B = regulator B, C = regulator C, DE = regulator D and E, FH = regulator F and H, ALL = all regulators.

- *Show break* command – The *show break* command causes the contents of the and-break table and the or-break table to be displayed on the console terminal.

  Syntax: SHOW⟨SP⟩BREAK⟨CR⟩

- *Show define* command – The *show define* command causes the symbol names used to construct the visibility register to be displayed on the console terminal.

  Syntax: SHOW⟨SP⟩DEFINE⟨SP⟩{reg-name, hex-id}⟨CR⟩

  – reg-name: the name of the visibility register defined within the HEX command set or by the *trace define* command.

  – hex-id: a 16-bit hexadecimal number of the hardware visibility-bit address or a software defined numerical tag for the visibility register.

- *Show name* command – The *show name* command displays the V$xxxxx symbol of the register, id, or signal name. If a register name is specified, the size of the register (number of V$ terms defined) is also displayed.

  Syntax: SHOW⟨SP⟩NAME⟨SP⟩{ident}⟨CR⟩

  – ident: symbol-name = the V$xxxx visibility symbol assigned by the appropriate .CDF data file, reg-name = the name of a register within the HEX command set or by the *define trace* command, hex-id = 16-bit SD bus or register ID, "signal-name" = the name of a visibility signal.

- *Show register* command–The *show register* command provides a console display of all the visibility registers that are defined.

  Syntax: SHOW⟨SP⟩REGISTER⟨CR⟩

- *Show trace* command–The *show trace* command provides a console display of the registers and signals currently selected for tracing and the current breakpoint options. The trace list can be altered by the trace command.

  Syntax: SHOW⟨SP⟩TRACE⟨CR⟩

- *Statestep* command–The *statestep* command causes the CPU clock to step a specified number of CPU clock cycles. If the SBI adapter visibility module is included, the *statestep* command will attempt to step the SBI clock one-half the number of cycles specified for the CPU clock.

  Syntax: STATESTEP⟨SP⟩[hex-num]⟨CR⟩

  –hex-num: a hexadecimal number of desired clock cycles to be executed in the range of 1 to 400.

- *T micro* command–The *T micro* command initializes the trace facility and causes one CPU clock cycle to be performed until a trace breakpoint condition occurs or until the step-count expires. The state of the CPU is then displayed on the console terminal in accordance with the current trace settings. If the SBI adapter visibility module is included, this command will attempt to step the SBI clock along with the CPU clock.

  Syntax: TMICRO⟨SP⟩[hex-num]⟨CR⟩

  –hex-num: a hexadecimal number of the clock steps to be executed.

- *T state* command–The *T state* command initializes the trace facility and causes one CPU-clock phase to be performed until a trace breakpoint condition occurs or until the step-count expires. The state of the CPU is then displayed on the console terminal in accordance with the current trace settings. If the SBI adapter visibility module is included, this command will attempt to step the SBI clock along with the CPU clock.

  Syntax: TSTATE⟨SP⟩[hex-num]⟨CR⟩

  –hex-num: a hexadecimal number of the clock phases to be executed.

- *Trace add* command–The *trace add* command is used to enter additional information to the register lists and signal trace lists. The register names and symbols can be intermixed in the command. The items in the command line are separated by spaces and the line cannot exceed 80 characters.

  Syntax: TRACE⟨SP⟩ADD⟨SP⟩[item]⟨CR⟩

  –item: reg-name = the names of the registers to be entered, symbol = the V$nnnn symbol that identifies the signal to be entered.

- *Trace define* command–The *trace define* command is used to define visibility registers in addition to those previously defined.

  Syntax: TRACE⟨SP⟩DEFINE⟨SP⟩⟨reg-name⟩⟨CR⟩

  –reg-name: the name assigned to the user visibility register.

- *Trace delete* command–The *trace delete* command is used to delete user-defined visibility registers that were established by the *trace define* command.

  Syntax: TRACE⟨SP⟩DELETE⟨SP⟩⟨reg-name⟩⟨CR⟩

  –reg-name: the name of the previously defined user visibility register.

- *Trace remove* command–The *trace remove* command allows register and signal information to be removed from the trace lists. The list of items in the command line to be removed must be separated by spaces and the line cannot exceed 80 characters including spaces.

  Syntax: TRACE⟨SP⟩REMOVE⟨SP⟩⟨item⟩⟨CR⟩

  –item: R* removes all registers from the list, S* removes all signals from the list, reg-name = the name of the register to be removed, V$nnnn = the symbol of the signal to be removed.

- *Trace restore* command–The *trace restore* command restores the default trace lists, including the visibility registers and signals.

  Syntax: TRACE⟨SP⟩RESTORE⟨CR⟩

VAX 8600 System Initialization

Before the VAX 8600 can be used to perform the customer's application, both the console subsystem and the CPU must be properly initialized. After the initialization sequence is started, the actions that occur depend on the position of the *local/remote* switch and *boot/halt* switch on the control panel. The console subsystem initialization process is initiated by one of the following events:

- When power is initially applied and the operator sets the *local/remote* switch to any position except the *off* position
- When power is restored after a system power failure
- When the processor initiates a reboot request to the console subsystem
- When the operator executes a *reboot* command

During a *reboot* command from the CPU or operator when the *boot/halt* switch is in the *halt* position, the following message will be displayed on the console terminal.

```
)))reboot
VAX 8600 Console V(n) (see note 1)
Initializing
Initializing power system
Initializing CPU
Total memory available is (nn) megabytes (see note 2)
)))
```

- note 1: (n) is the version number of the console subsystem software.

  note 2: (nn) is the value of the total number of Mbytes. The message that follows this message indicates the number of Mbytes in each array that is installed in the eight CPU memory slots, and the status and revision level of the SBIA adapters that are installed in the system.

Control is then transferred to the console subsystem PROM by any of the events that caused the system initializing process. The console PROM code executes a series of self-tests, loads the bootstrap program from the RL02 disk drive, and allows the console microprocessor to execute the bootstrap program. The bootstrap program loads the console bootstrap program (EDOAA) into the microprocessor RAM, which then performs the following operations:

- Initializes the console subsystem and power system
- Initializes the macro context in the console subsystem
- Initializes the CPU to perform the macrocode
- Monitors the status of the control panel switches

- Console Program Initialization

  The console program controls the operation of the console subsystem and the CPU. The following functions are performed during the console program initialization:

  - The console program banner and revision number are displayed.
  - The microprocessor vectors and internal data structures are initialized.
  - The microprocessor is initialized and allowed to access the 256-Kbyte RAM.
  - The *Initializing* message is displayed on the console terminal to indicate the initializing sequence is in process.
  - The SD bus signal name tables and the console subsystem microcode are loaded from the RL02 disk drive into console subsystem RAM.

- The power system is initialized and the *Initializing power system* message is displayed on the console terminal.
- The clocks in the CPU are initialized.
- The console subsystem is initialized to the macro context.

- POWER SYSTEM INITIALIZATION

The power system initialization, performed during the console program initialization, sets the power supply regulators to the proper output voltages and enables the environmental monitoring module (EMM) to report environmental conditions to the console subsystem. During the power system initialization, the following operations are performed:

- The proper parameters are set in the console subsystem serial-line interface that communicates with the EMM.
- The communication of the EMM channel is verified.
- The temperature and voltage parameters are loaded in the EMM.
- The EMM functions are enabled.
- The power supply regulators are enabled and their operation is verified.
- The EMM magnetic disk display is reset to clear any error indications.

- MACRO CONTEXT INITIALIZATION

The macro context initialization is automatically performed following the console program initialization. During the initialization sequence, control is transferred to the macro control program overlay, which initializes the CPU and provides control to the macro command set.

- CPU INITIALIZATION

When the console subsystem enters the macro context, the console program has access to the macro command set to initialize the CPU for microcode operation. The console program submits the LOAD.CMD and the ULOAD.COM indirect command files internally. These files contain the commands necessary to initialize the CPU by loading the control RAMs in the processor from files on the RL02 drive. When the CPU initialization has been completed, the console program tests the state of the *boot/halt* switch on the control panel to determine if a warm-restart or cold-restart process is to be performed. A message is displayed on the console terminal indicating which process is being performed.

If the *boot/halt* switch is in the *restart boot* or *restart halt* position, or the *local/remote* switch is in the *local disable* position, a warm-restart can occur. The *Attempting System Restart* message is displayed, the general purpose registers in the CPU are initialized, the warm-start flag in the restart parameter block (RPB) is set, and the CPU begins operation at the restart address specified by the RPB. When the CPU initialization is completed, the console subsystem enters program I/O mode.

If the *boot/halt* switch is in the *restart boot* or *boot position* and the *local/remote* switch is not in the *local disable* position, a cold restart will be initiated. The *Attempting System Initialization* message is displayed, the operating system is reloaded from the system disk and self-started, and the program I/O mode is enabled. The cold-start procedure is controlled by the DEFBOO.COM command file on the RL02 disk drive. The file initializes the CPU and I/O adapters, locates a block of valid memory, and loads the VBM program from the RL02 drive. The VBM program loads the operating system from the system disk drive, initiates the program operation, and transfers control to program I/O mode. If a failure occurs during the cold-start procedure, the console program enters the macro context in console I/O mode.

If the *boot/halt* switch is in the *halt* position, the macro context of the console I/O mode is entered after the macro context initialization is performed.

Chapter 9 · Computer Bus Interconnects

The input/output (I/O) subsystems of the VAX-11/750 processor, VAX-11/780 series processors, and VAX 8600 processor connect to an internal bus structure, which forms the communication path between the CPU and memory, and between the CPU and devices connected to the I/O subsystems. The internal bus is a fast and reliable synchronous path for the transfer of information. The bus structure for VAX-11/750 processor is the CPU/memory interconnect (CMI). For the VAX-11/780 series and VAX 8600 processors, the bus structure is the synchronous backplane interconnect (SBI). Bus information is transferred to all the units on the bus that are designated as a nexus. Each nexus can initiate information transfer by decoding the bus information to determine the type of transaction required. The computer bus interconnects transfer address, data, and control information between the processor and devices, perform priority arbitration among the CPU and devices, and translate the addresses between the processors, memory, and devices. Some of the features and functions of the computer bus interconnects are as follows:

- Transactions are interleaved to allow several simultaneous operations to be performed during the transfer process.
- Priority arbitration logic monitors the arbitration lines of each nexus.
- Failure of one nexus does not inhibit the operation of the remaining units.
- Parity and protocol checking of the information transferred through the interconnects insures that the data received is the same as the data transmitted. Errors in the protocol of a transaction are recorded.
- Transactions performed during the most recent bus cycles are recorded and stored for analysis.

· CPU/Memory Interconnect

The CPU/memory interconnect (CMI) is the internal bus for the VAX-11/750 processor. It is a synchronous, interlocked, backplane bus that connects the internal logic of the processor to the main memory and to the I/O subsystem through the processor backplane. The CMI is also the internal processor path for the remote diagnostic logic, UNIBUS interface logic, writable control store logic, and the memory interconnect and controller.

CMI Signal Lines

The CMI bus consists of 45 bidirectional lines used to transfer address, data, control, status, and priority arbitration signals. A unidirectional bus clock line provides the signals used to synchronize the transfer of information. A master/slave relationship exists between the units on the CMI so that the nexus that has control of the bus is considered the master and the receiving nexus is designated as the slave. Figure 9-1 shows the CMI signal lines and table 9-1 lists their functions.

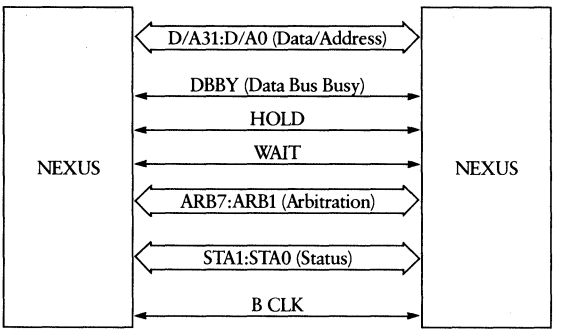


Figure 9-1 ▪ CMI Signal Lines

Table 9-1 ▪ CMI Signal Line Functions

Line	Description
D/A31:D/A0	Data/address–Multifunction lines used to transfer address, data, and control information between the CPU and a nexus.
DBBZ	Data bus busy–Asserted by the master to specify that an address is contained on the lines D/A31:D/A0.
Hold	Asserted to temporarily suspend the activity on the CMI.
Wait	Asserted by an I/O subsystem to initiate a processor interrupt.

Line	Description
ARB7:ARB1	Arbitration – The priority arbitration lines assigned to the I/O subsystems as follows: **Line* I/O Subsystem** ARB7 Remote diagnostic module (RDM) ARB6 Reserved ARBR Reserved ARB5 UNIBUS adapter 0 ARB4 UNIBUS adapter 1 ARB3 MASSBUS adapter 0 ARB2 MASSBUS adapter 1 ARB1 MASSBUS adapter 2
	* Line ARB7 has the highest priority and the CPU, which has no line, is assigned the lowest priority.
STA1:STA0	Status – Two lines that transfer signals by the slave to indicate the following status conditions to the master. **Status Line 1 0** / **Condition** 0 0 / No response – Master attempted to access nonexistent memory for a read or write operation
	0 1 / Data returned to master contains an uncorrectable error
	1 0 / Data transmitted is corrected
	1 1 / Data transmitted is correct
B CLK L	Timing – A timing signal generated by the CPU to synchronize the system activities.

Transfer Format

Information is transferred on the data and address lines in two operations. A master gains access to a slave by transmitting the physical longword address of the slave and by asserting the data-bus-busy signal for one bus-clock cycle. A longword is then transferred to or from the slave. If the slave is not ready to accept the information, it asserts the data-bus-busy line until it is ready. Figure 9-2 shows the CMI address format and figure 9-3 shows the CMI data format.

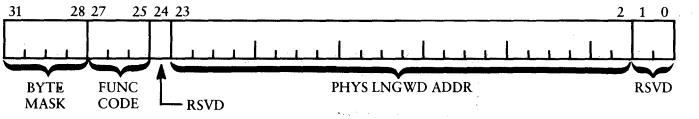


Figure 9-2 ▪ CMI Address Format

Bit	Function
31:28	BYTE MASK – Set to designate which bytes are valid for transfer as follows: **Bit** **Valid** 31 BYTE 0 30 BYTE 1 29 BYTE 2 28 BYTE 3
27:25	FUNC CODE (Function code) – An octal value used to designate the operation being performed by the master as follows: **Value** **Operation** 0 Read 1 Read lock 2 Read with modify intent 3 Undefined 4 Write 5 Write lock 6 Write vector 7 Undefined
24	RSVD (Reserved).
23:2	PHYS LNGWD ADDR (Physical longword address) – The physical longword address of data to be read or written.
1:0	RSVD (Reserved).

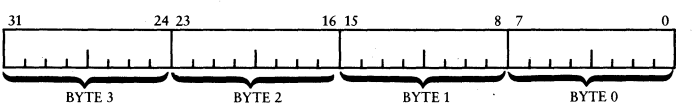


Figure 9-3 ▪ CMI Data Format

Bit	Function
31:24	BYTE 3 – Contains the byte 3 value of the data
23:16	BYTE 2 – Contains the byte 2 value of the data
15:8	BYTE 1 – Contains the byte 1 value of the data
7:0	BYTE 0 – Contains the byte 0 value of the data

CMI Operation

A minimum duration of three clock cycles is normally required to transfer one longword of data during a CMI read or write operation. The time required for a transfer depends on when a slave device returns data or status information. The cycles used during a transfer are the arbitration cycle, the CMI address cycle, and the CMI data cycle. During the arbitration cycle, the CMI is idle and the data-bus-busy and hold signals are not asserted. During the address cycle, the CMI address and the data-bus-busy signal are asserted by the master. During the CMI data cycle, the data-bus-busy signal is asserted by the slave if it is not ready to complete the transactions. If a slave is ready to receive data, it does not assert a data-bus-busy signal; therefore, only two cycles are required for the transaction.

The data cycle can be a read or write transaction. In a read transaction, the slave asserts the data-bus-busy signal and then transmits the requested data and data status information. In a write transaction, the slave clocks the data, deasserts the data-bus-busy signal, and returns status information.

A nexus can assert its arbitration signal at any time. Arbitration occurs when the data-bus-busy and hold signals are not asserted. The nexus with the highest priority arbitration level asserted inhibits the lower priority subsystems from gaining access to the bus. On the next positive transition of the clock cycle, the new master asserts both the physical longword address of the slave and the data-bus-busy signal. All other units recognize that an address longword is on the CMI, and the addressed slave responds accordingly.

CMI Physical Address Space

The assignment of the physical addresses space for the CMI is shown on the physical address map in figure 9-4. The address space includes map registers, internal and external MASSBUS and UNIBUS registers, and control and status registers.

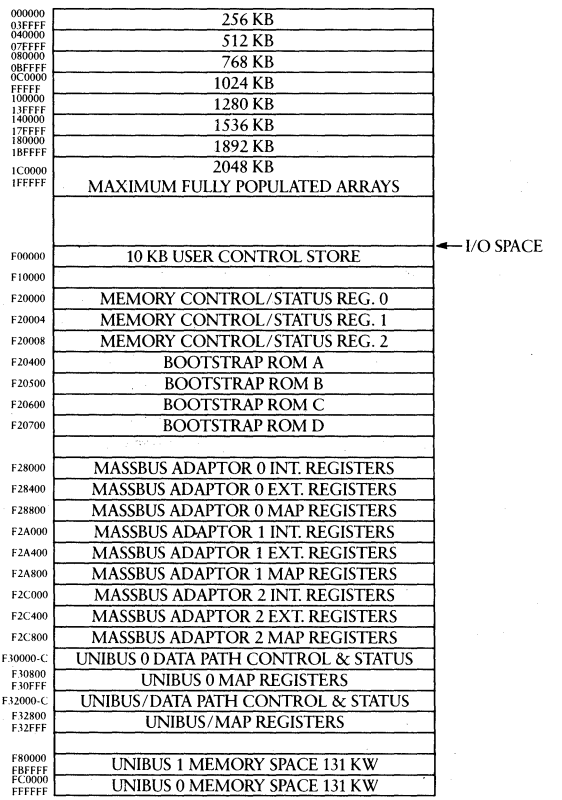


Figure 9-4 • CMI Physical Address Space

• Synchronous Backplane Interconnect

The synchronous backplane interconnect (SBI) bus is the data path that connects both the CPU and main memory of the large VAX systems to the

I/O subsystem adapters. All UNIBUS, MASSBUS, CI bus, and DDI bus devices communicate with the processor through the SBI bus. Figure 9-5 shows the basic configuration of the devices and adapters to the SBI. In the VAX-11/780 series processors, the memory controllers connect to the SBI bus and all memory transactions are performed on the SBI. In the VAX 8600 processor configuration, the SBI adapters connect to the Mbox through the A bus and the Mbox communicates with the main memory. The SBIA 0 adapter is included with the processor and the SBIA 1 adapter can be added to the system as an option. The SBI bus provides the following features:

- Transfers 64-bit data in two consecutive cycles.
- Permits a throughput rate of up to 13.3 million bytes per second and a bus cycle time of 200 nanoseconds.
- Performs distributed arbitration to increase transfer speed.
- Includes a 16-level silo register that monitors the SBI operations and maintains a history of the 16 most recent cycles of bus activity.
- Includes maintenance registers to help in determining the cause of bus-specific errors.

SBI Signal Lines

The SBI bus contains 84 signal lines that connect to each nexus. The bus consists of arbitration, information, response, interrupt, and control lines as shown in figure 9-6. The maximum physical length of the SBI bus is 3 meters (9.8 feet). Table 9-2 lists the signals on the lines and defines their function.

Table 9-2 • SBI Signal Line Functions

Line	Description
TR15:TR0	Transfer Request – Sixteen arbitration lines that establish a fixed priority to each nexus. Line TR0 is the highest priority and line TR15 is the lowest. Line TR0 is used during the SBI unjam operations. For the VAX 8600 CPU, line TR1 is asserted to indicate that a DMA transfer from memory to a device is requested and line TR2 is the CPU priority. The line assignments are as follows: **Line** **Function** 0 Hold (SBI unjam) 1 Memory controller 1 (VAX 8600: DMA data request from memory)

Line	Description
	Line **Function** 2 Memory controller 2 (VAX 8600: CPU priority) 3 UNIBUS adapter 1 4 UNIBUS adapter 2 5 UNIBUS adapter 3 6 UNIBUS adapter 4 7 Reserved 8 MASSBUS adapter 1 9 MASSBUS adapter 2 10 MASSBUS adapter 3 11 MASSBUS adapter 4 12-15 Reserved
B31:B0	Information–Multifunction lines that contain the address, data, or function information.
P1:P0	Parity–These lines provide even parity for detecting single bit errors as follows: **Line** **Function** P0 Parity for the tag, identifier and mask fields
	P1 Parity for the information field
TAG2:TAG0	These lines contain an octal code from the transmitting nexus to define the type of information on the information lines as follows: **Value** **Definition** 0 Read data 3 Command address 5 Write data 6 Interrupt summary read
ID4:ID0	Identifier–These lines contain a source or destination code that identifies the logical source or destination of the information contained on lines B31:B0. The ID codes are assigned to a commander or responder nexus and the identifying code is the same as defined for priority assignments of the TR15:TR0 lines.
M3:M0	Mask–These lines are encoded by the transmitter to specify the conditions and operations of the information on lines B31:B0.
Fault	This line is asserted by a nexus to indicate that an error has been detected in the SBI protocol or that a malfunction has occurred in the information path.

Line	Description
CNF1:CNF0	Confirmation – A binary code that informs the transmitter of the condition of the information received as follows: **Code** **Definition** 0 No response to a commander's selection 1 A positive acknowledgement to a transfer 2 The busy response to a command/address transfer from a nexus that is presently unable to execute the command 3 The error response to a command/address transfer from a nexus that cannot perform the command
Unjam	This line is asserted by the console to initialize all of the nexuses to a defined state.
Fail	This line is asserted when the power supply AC LO signal to the nexus is asserted and inhibits the CPU from initiating a powerup service routine.
Dead	This line is asserted by the CPU clock module or terminating networks to indicate that a power failure is impending and prevents invalid data from being received by a nexus when the bus is in an unstable state.
Interlock	Except for the VAX 8600 processor, this line is asserted by a memory controller during an interlock read mask or interlock write mask function to insure exclusive access to a specific memory location.
Clock	Six lines from the CPU clock module that provide clock signals to synchronize the information transfers on the SBI.
REQ7:REQ4	Request – Four lines assigned to each nexus that is capable of interrupting the processor. The lines are priority-encoded in an ascending order, with REQ4 as the highest priority.
Alert	This line is asserted by each nexus that does not have interrupt capability and informs the processor of a change in status, power conditions, or operating environment.
MP1 2	Not used.

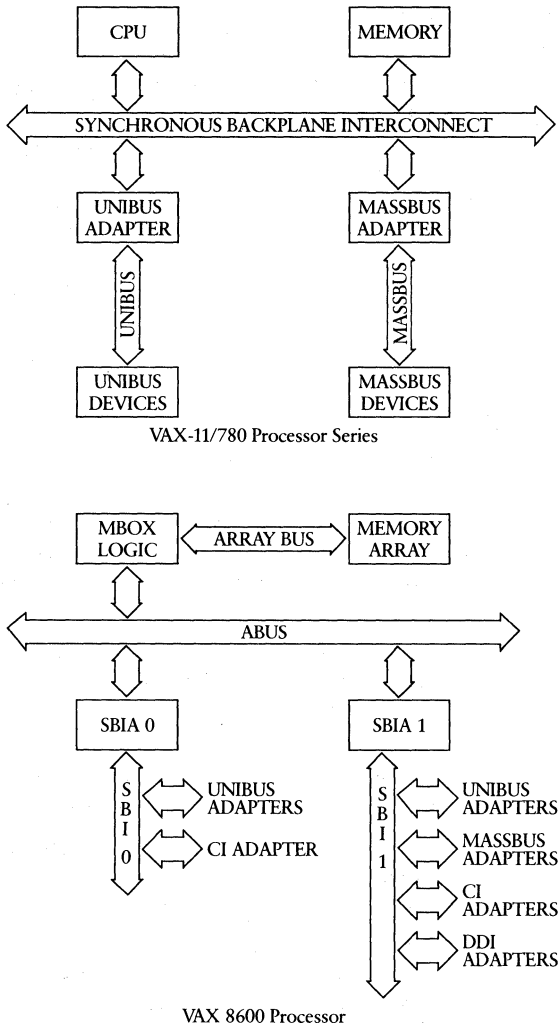


Figure 9-5 ▪ SBI Basic Bus Configuration

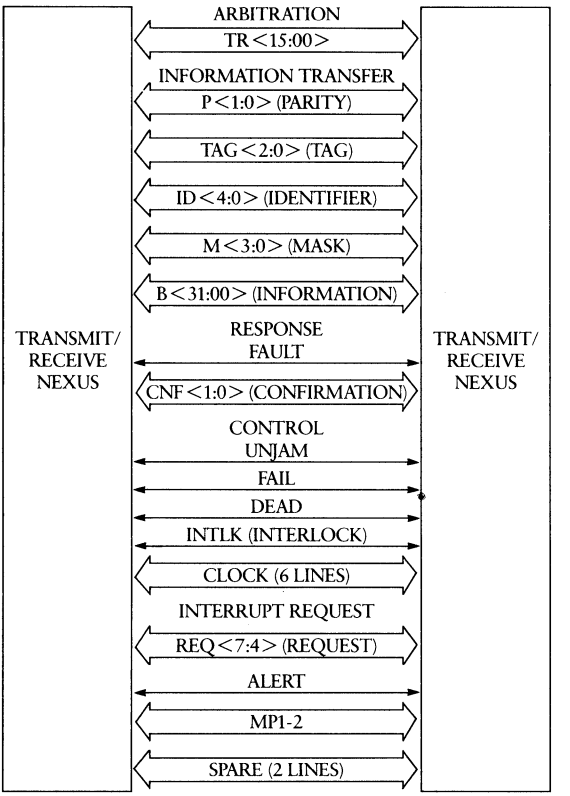


Figure 9-6 • SBI Signal Lines

Sixteen transfer request lines (TR15 through TR00) are included to establish the access priority and one line is assigned to each nexus. Line TR0 is assigned the highest access priority and TR15 is the lowest. To acquire control of the information path, a nexus asserts its transfer request line at the beginning of the cycle. At the end of the cycle, the nexus examines the state of all the transfer request lines that are assigned a higher priority. If a higher priority nexus is not requesting control, the operating nexus removes its request and asserts the information summary signals. In the VAX-11/780 series processors, the CPU is assigned the lowest priority and does not require a transfer line.

The information transfer group consists of 46 lines used to exchange command/address and summary information. They include the parity,

information tag, source and destination identity, mask function and information lines. Each exchange consists of from one to three transfers.

The response group consists of three lines that contain error information and information related to the condition of the data transferred.

The control group consists of 10 lines that control the information transfer operations. Six of the lines contain the clock signals used to synchronize the information transfers. The control lines synchronize the system activities and provide specialized system communications. The lines provide initialization, powerfail, and restart functions for the system. In addition, a path is provided for coordinating memories to assure access to shared structures. The clock signals provide a universal time base for all nexus operations. All SBI clock signals are generated on the CPU clock module and have a 200-nanosecond cycle.

The interrupt request group consists of four interrupt request lines that are assigned to each nexus and an alert line that indicates a change in status power conditions or operating environment.

SBI Operation

Each nexus can transmit information to the SBI and receive information from it. It also performs priority arbitration to gain access to the SBI. The communication protocol allows the information path to be time-multiplexed so that up to 32 data exchanges can occur simultaneously. The SBI provides checked, parallel information transfers that are synchronized to the system clock. A nexus can perform interconnect arbitration and information transfer during each clock period or cycle of 200 nanoseconds to achieve a maximum transfer rate of 13.3 Mbytes per second. The nexus can perform the following functions:

- As a *commander,* it transmits command and address information.
- As a *responder,* it recognizes command and address information directed to it and responds to that information.
- As a *transmitter,* it controls the signal lines.
- As a *receiver,* it samples and examines the signal lines.

When the CPU issues a read command, the CPU assumes command because it issues command/address information to the SBI. It also operates as a transmitter because it controls the signal lines. When the device acting as a responder returns the requested data, the CPU performs the function of a receiver because it examines the signal lines.

Except for the VAX 8600 processor, when a memory read transfer occurs, the memory is the responder and receiver because it recognizes and responds to command/address information and examines the signal lines. When memory returns the requested data, it operates as a transmitter. In

the VAX 8600 processor, the memory subsystem communicates through the A bus and the SBIA performs all the necessary nexus functions for the SBI bus.

Two or three successive SBI cycles are required for a write operation and two successive cycles are used for the extended read operation and for an interrupt read summary exchange.

To acquire control of the SBI bus, a nexus asserts its transfer request line at the beginning of a clock cycle. At the end of a cycle, the nexus examines the state of all the transfer request lines that have a higher priority. If a nexus with a higher priority is not arbitrating for control of the SBI bus, the controlling nexus removes its transfer request signal and asserts the information line signals. The CPU is assigned the lowest priority for line arbitration and does not require a transfer request signal. It gains control of the SBI bus by default when no other nexus is arbitrating.

SBI Physical Address Space

The user has access to the SBI physical address space through a 30-bit physical byte address. Because device registers are accessible only as longword addresses, the 30-bit physical byte address is converted to a 28-bit SBI longword address through a translation buffer. The process is shown in figure 9-7. The 28-bits of the address defines a longword address space in the SBI that is grouped into a memory space and I/O register space. Bits 1 and 0 are used to align the data on a read operation. Figure 9-8 shows the SBI address space assignments; both physical and SBI addresses are provided.

Information Transfer

Command/address, data, or interrupt summary information is transferred during the information transfers. During write operations, the commander uses two or three successive SBI cycles, depending on the number of data words to be written in the exchange. For one data word, the commander transmits the command/address in the first cycle and a data word in the second cycle. For a two-word transfer, the commander transmits the command/address in the first cycle, the first data word in the second cycle, and the second data word in the third cycle. Figure 9-9 shows the formats of the information transferred.

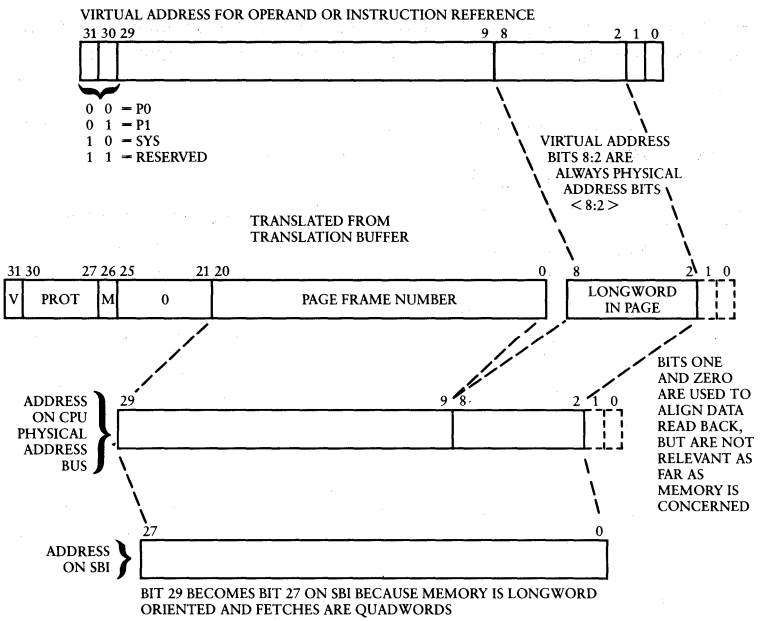


Figure 9-7 ▪ Physical to SBI Address Translation

Read commands are also initiated with a command/address transmitted from the commander. Because the data originates at the responder, the request data may be delayed by the characteristic access time of the responder. As in a write operation, the read operation transmits data using one or two successive cycles, depending on whether one or two data words were requested.

An interrupt summary exchange is the response to a device-generated interrupt to the CPU. The exchange is initiated with an interrupt-summary-read transfer from the CPU. The exchange is completed two cycles later with an interrupt-summary-response transfer containing the interrupt information.

- PARITY FIELD

The parity field provides even parity for detecting single-bit errors in the information. A transmitting nexus generates parity for the tag, identifier, and mask fields and parity for the information field. The P0 and P1 signals are generated as the sum of all logical one bits, including the parity bit in the checked field. When the SBI bus is idle, it assumes an all-zero state.

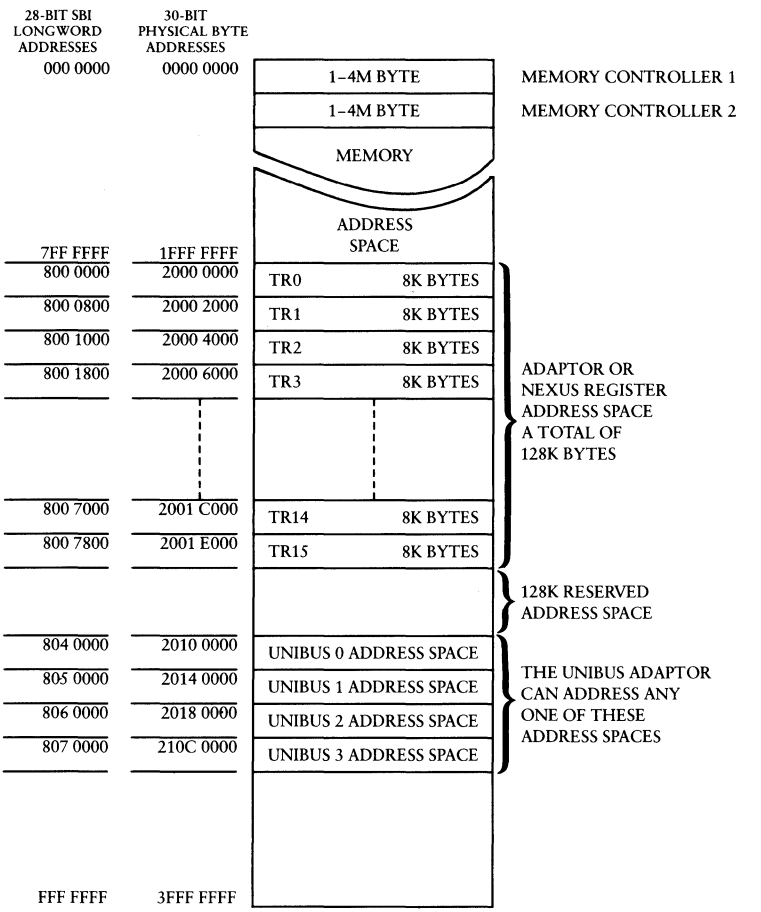


Figure 9-8 • SBI Physical Address Assignments

- TAG FIELD

The transmitting nexus asserts the tag field to specify the type of transaction to be performed and to determine the interpretation of the information on lines B31:B0 and the identifier field. The tag field is also used in conjunction with the mask field to define special read and write data conditions. The command/address tag specifies that the data lines contain a command/address word and that the identifier field contains a unique code to identify the source (commander) of that command. During a write command, the identifier field indicates the source of the data and the

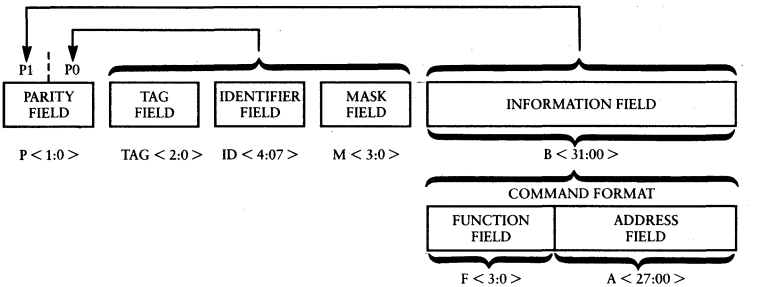


Figure 9-9 ▪ Information Transfer Formats

address information specifies the location where the data is to be written. For a read command, the identifier code represents the destination of the data at the location specified in the address lines. The octal code of the tag field and the type of data being transmitted are defined in table 9-3.

Table 9-3 ▪ Tag Field Assignments

Code	Transaction
000	Read data
001	Reserved
010	Reserved
011	Command or address
100	Reserved
101	Write data
110	Interrupt summary read
111	Reserved

▪ IDENTIFIER FIELD

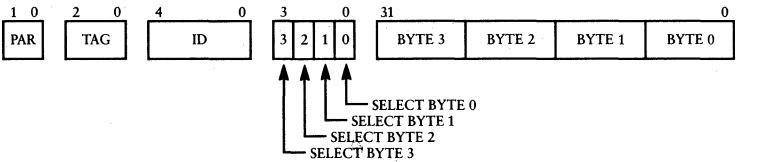
The identifier (ID) field contains a hexadecimal code that identifies the logical source or destination of the information contained in the lines B31:B0. The code corresponds to the priority assignments on the TR15-TR0 lines. For example, code 01 is assigned to memory controller 1. For the VAX 8600 processor, code 01 indicates that the memory is requesting the SBI bus for a DMA transfer to a device. The VAX 8600 requests the SBI bus by asserting line TR2; however, code 16 is used to specify the request. The identifier field assignments are listed in table 9-4.

Table 9-4 • Identifier Field Assignments

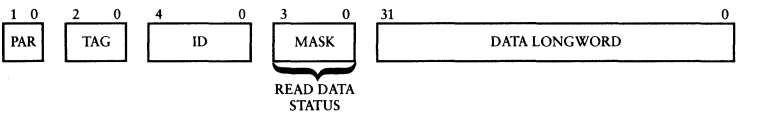
Code	Assignment
00	Hold
01	Memory controller 1 (VAX 8600: DMA request from memory)
02	Memory controller 2 (not assigned on the VAX 8600)
03	UNIBUS adapter 1
04	UNIBUS adapter 2
05	UNIBUS adapter 3
06	UNIBUS adapter 4
07	Reserved
08	MASSBUS adapter 1
09	MASSBUS adapter 2
10	MASSBUS adapter 3
11	MASSBUS adapter 4
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	CPU

- MASK FIELD

The functions performed by the mask field depend on the type of transaction that was selected. The mask is used with the read data, write data, and command/address information as specified by the tag field. During a command/address or write data operation, the mask field specifies the number of bytes on lines B31:B0 to be interpreted. During a read data operation, the mask field indicates the status of the data on lines B31:B0. The mask field contains all zeros to indicate an information-summary read operation. Figure 9-10 shows the mask field formats for the command/address or write data operations and the status format for a read data operation. The mask field values are described in the command described in the following paragraphs.



Command/Address or Write Data Format



Read Data Format

Figure 9-10 ▪ Mask Field Formats

▪ FUNCTION FIELD

The command/address format contains a byte mask, function code field, and a 28-bit physical address. The function codes are provided by information lines B31:B28 and are listed in table 9-5. If line A27 of the address information is cleared, the address is in main memory, and if line A27 is set, the address is in the I/O space.

Table 9-5 ▪ Command/Address Function Codes

Function Code *	Command
0001	Read masked
0010	Write masked
0100	Interlocked read masked
0111	Interlocked write masked
1000	Extended read (write mask ignored)
1011	Extended write masked

* All other function codes are reserved and the write mask is ignored.

▪ TRANSFER FORMATS

The four basic types of information transfers are read data, command/address, write data, and interrupt summary. The type of transfer is specified in the tag field of the command information.

Read Data Tag A read data transfer is specified by a tag field value of 000 and indicates that the information field (B31:B0) contains the data requested by a previous command. The status of the retrieved data can be one of three types – read data, corrected read data, or read data substitute – and is identified by the mask field. The read data normally contains no errors; however, if an error is detected, the condition of the data is identified. The destination of the data is specified by the identifier field. Figure 9-11 shows the format of the read data information and table 9-6 lists the mask codes and read data status.

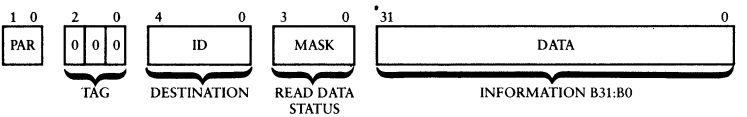


Figure 9-11 ▪ Read Data Formats

Table 9-6 ▪ Read Data Mask Assignments

Mask				Status
M3	M2	M1	M0	
0	0	0	0	Data is correct
0	0	0	1	Data has been corrected
0	0	1	0	Data has uncorrectable error

Command/Address Tag A command/address transfer is specified by a tag field value of 011 and indicates that the data lines contain a command/address word. Six types of command/address transfers can be performed: three read transfers and three write transfers. The information field (B31:B0) contains a 4-bit function identifier (F3:F0) that specifies a read or write sequence and an address location (A27:A0) that indicates where the data will be read or written. The identifier field contains the logical destination of the data for a read data transfer and the logical source of the data for a write data transfer. The contents of the mask field depends on the

type of transaction being performed. Figure 9-12 shows the format of the information for command/address transfer. Table 9-7 defines the transfer for each function code.

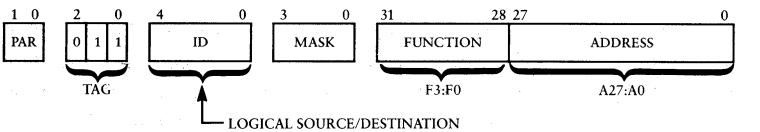


Figure 9-12 ▪ Command/Address Format

Table 9-7 ▪ Command/Address Functions

Mask	Function				Transfer
	F3	F2	F1	F0	
Number of bytes	0	0	0	1	Read masked format – Data is read at the specified address
Number of bytes	0	1	0	0	Interlock read masked format – Data is read at the specified address
All zeros	1	0	0	0	Extended read format – Data is read at the specified address
Write data longword	0	0	1	0	Write masked format – Data is written to the specified address
First write data	0	1	1	1	Interlock write masked longword format – Data is written at the specified address
First write data	1	0	1	1	Extended write masked format longword – Data is written at the specified address

Bits A27:A0 define the longword address space, which is grouped into two sections. Addresses 0 through 7FFFFFF (hexadecimal) are reserved for primary memory and 8000000 through FFFFFFF (hexadecimal) are reserved for device addresses. Each nexus is assigned a 2,048, 32-bit longword address spaces for control and status. The addresses assigned are determined by the TR field, which specifies one of the address space block for

one of the 16 nexuses. Figure 9-13 shows the format of control address space information.

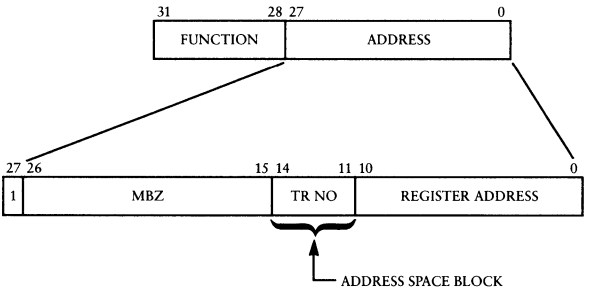


Figure 9-13 • Control Address Space Format

Write Data Tag A write data transfer is specified by a tag field value of 101 and indicates that the information on lines B31:B0 contains the data to be written at the location specified by the address field of the previous write command. The data is asserted in the SBI cycle immediately following the command/address cycle. The mask field is defined in table 9-8 and is used to select one or more bytes of data in the information field that are to be written. The ID field designates the commander that initiated the data transfer. The format for the write data operation is shown in figure 9-14.

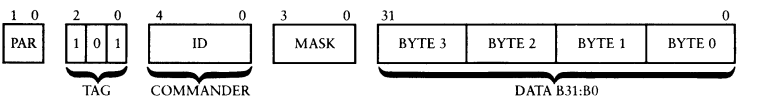


Figure 9-14 • Write Data Format

Table 9-8 • Write Data Mask Assignments

Bit	Selection
M3	Byte 3 of the information (B31:B24)
M2	Byte 2 of the information (B23:B16)
M1	Byte 1 of the information (B15:B8)
M0	Byte 0 of the information (B7:B0)

Interrupt Summary Tag The interrupt summary sequence consists of a transaction indicating the interrupt level being serviced and an identifying response from the requesting device. The format of the interrupt request and response is shown in figure 9-15. The interrupt request is specified by a tag field value of 110 and indicates that the information on lines B7:B4 is an interrupt level mask for the interrupt-summary read command. The interrupt level mask specifies the interrupt level being serviced as a result of the interrupt request. The ID field identifies the CPU as the commander. A tag field value of 000 is the response that identifies the device that is requesting the interrupt. The starting address of the service routine is determined by the identity and interrupt level. The vector generating pair of bits (B16 and B0) specifies the TR level of the nexus that requested the interrupt. Two bits are used to insure that the parity is even. The service routine is determined by the identity of the device and the interrupt level. The ID field specifies the destination of the response. All other tag values are reserved.

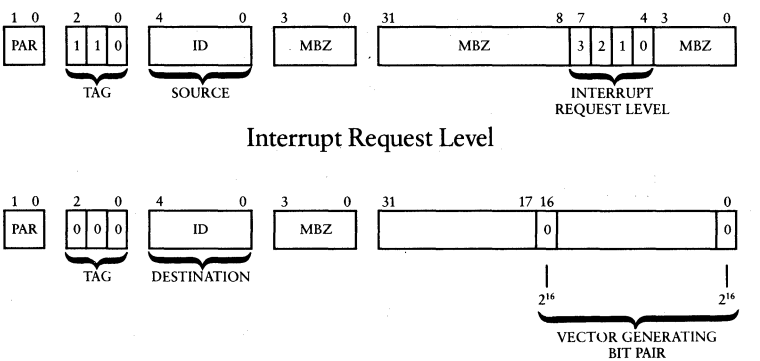


Figure 9-15 ▪ Interrupt Summary Format

Read Masked Function The sequence of the SBI read transaction is shown in figure 9-16. When a commander gains control of the SBI, it asserts the information lines. The command/address format instructs the addressed nexus to retrieve the addressed data word and to transfer it to the destination specified in the identifier field. Two SBI cycles after the assertion of the command/address, the addressed nexus responds to the transfer with an acknowledge message if it can perform the command.

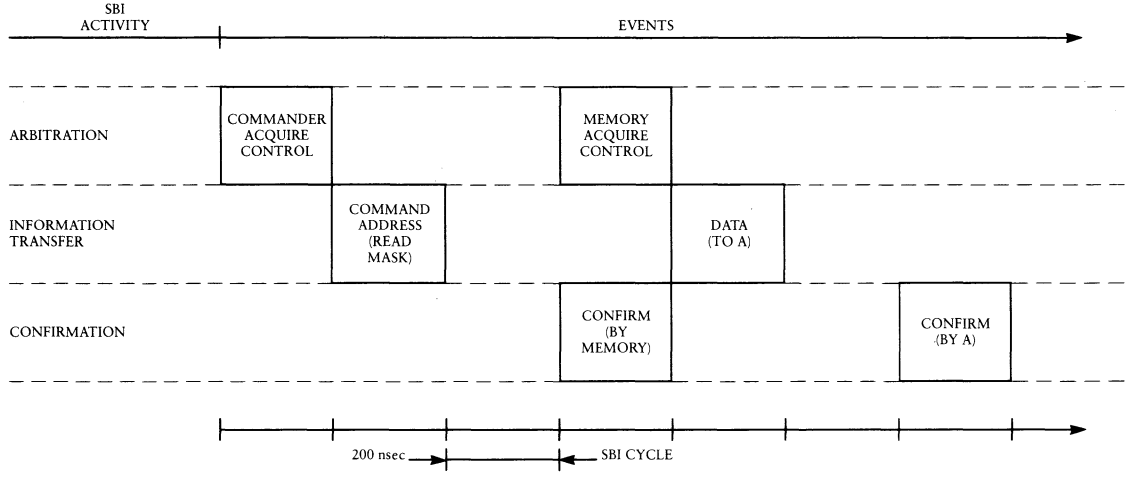


Figure 9-16 ▪ SBI Read Transaction

Write Masked Function The write masked function is used to modify stored data. It instructs the responder to modify certain bytes using data transmitted in the succeeding cycle. The bytes to be modified are specified by the mask signal. The memory or I/O address space is selected by line B27. The timing sequence for both a single SBI write transaction and two SBI write transactions is shown in figure 9-17.

Interlock-read-masked Function The interlock-read-masked function is used with the interlocked-write-mask command to insure exclusive access to a particular memory location. It causes the addressed nexus to retrieve and transmit the addressed data in a similar operation as the read masked function. The addressed memory controller asserts the interlock line and the nexus responds with a busy confirmation to interlock the read-masked commands. The interlock is cleared when the interlock-write-masked function is initiated.

Interlock-write-masked Function The interlock-write-masked function instructs the addressed nexus to modify the bytes specified by the mask field using the data transmitted in the succeeding cycle.

Extended Read Function The extended read function instructs the addressed nexus to retrieve the 64 bits of data selected by address bits A27:A0 and to transmit the data to the nexus specified by the identifier field in the command. The functions performed during this operation are similar to the read masked transfer functions. The responder transmits the data in two successive cycles with the low 32 bits preceding the high 32 bits. The mask code value transmitted from the commander is 0000. Figure 9-18 shows the extended read transactions to nexus A and B for the VAX-11/780 processor series. A single and dual memory controller system is shown. The separate memory controllers are designated M1 and M2. In the VAX 8600 processor, the operation is similar except that the SBI communicates with the SBIA. The SBIA then communicates with the Mbox which controls the memory.

Extended Write Masked Function The extended-write-masked function instructs the addressed nexus that 64 bits of data are to be written. Bits A27:A0 indicate the low 32-bit address. The data to be written is transmitted in two 32-bit words. The first word corresponds to A0 = 0 and the second word corresponds to A0 = 1. The mask field that accompanies the command address transmission indicates the number of bytes to be written in the first data word. The mask field that accompanies the first write data word transmission indicates the number of bytes to be written in the second write data word. The mask field of the second data word cycle is ignored by receivers but must be transmitted as zeros. The assertion of a particular mask bit signifies that the byte corresponding to that mask bit is to be modified. The component implementing the extended write masked function must implement all combinations of the mask signal.

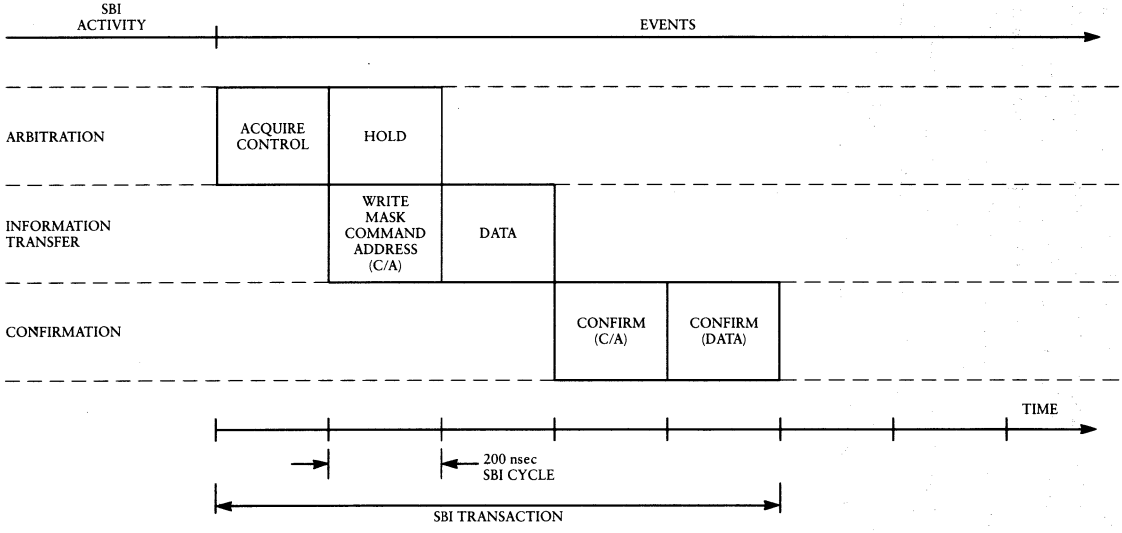


Figure 9-17 ▪ SBI Write Transaction

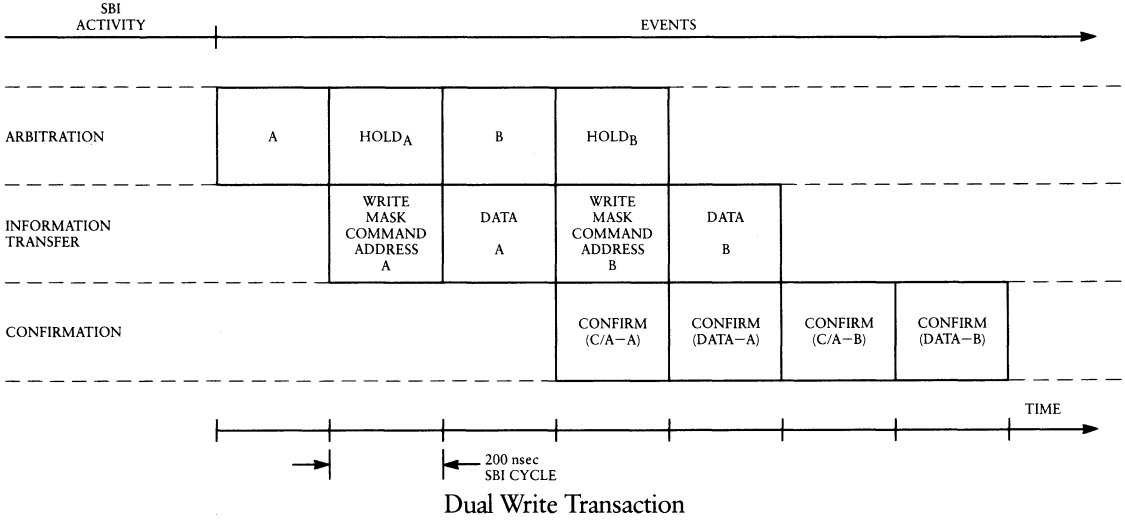


Figure 9-17 ▪ SBI Write Transaction

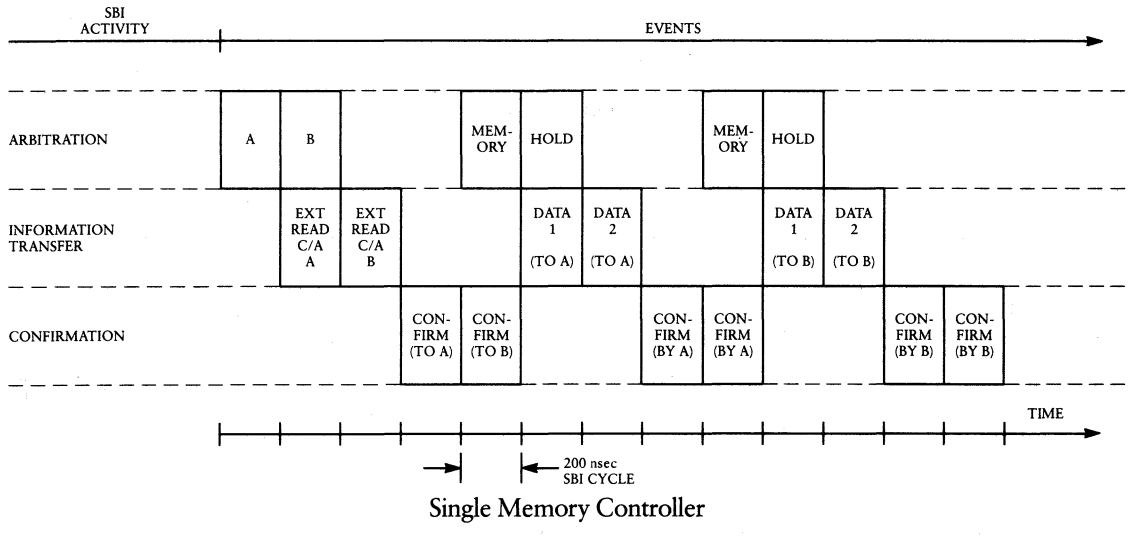


Figure 9-18 ▪ Extended Read Transactions

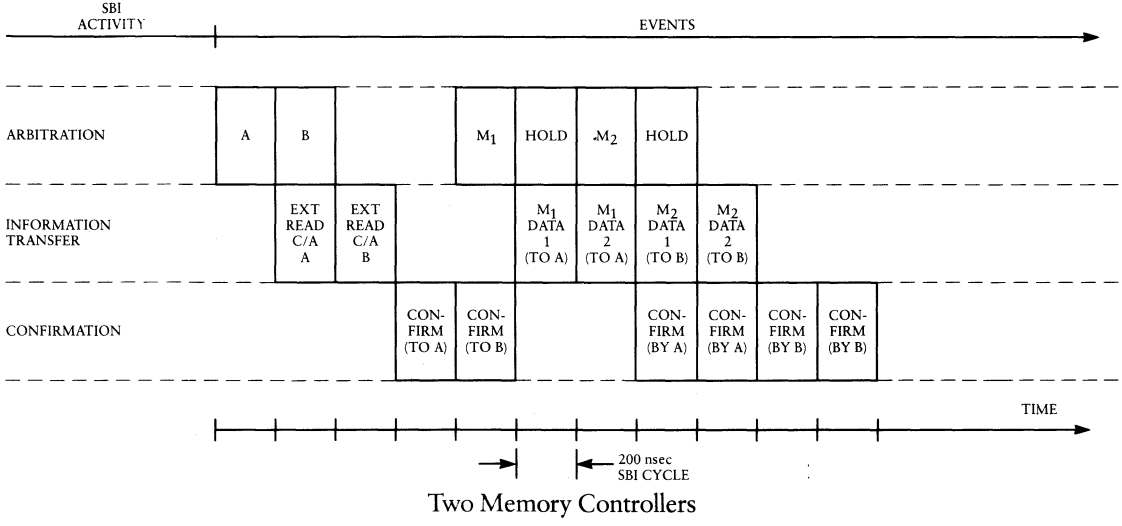


Figure 9-18 • Extended Read Transactions

Chapter 10 • VAX Privileged Registers

The VAX processor register space provides access to many types of CPU control and status registers such as memory management base registers, processor status longword, and the multiple stack-pointer registers. The VAX privileged registers are accessible only by move to processor register (MTPR) and move from processor register (MFPR) instructions, which are controlled by the kernel executive in the VAX/VMS operating system. The operating system manages these registers for the user; however, they are available to system programmers, operators, and maintenance personnel. The VAX processor register space includes architecturally defined registers that may be implemented in any VAX processor and registers that are included only in a specific type of VAX processor.

Architectural Processor Registers

Table 10-1 is a list of the architectural processor registers, the register address and type, and the VAX processors where the register is located.

Table 10-1 • VAX Architectural Processor Registers

Register Name	Acronym	Address	Type*	VAX** Processor
Kernel Stack Pointer	KSP	00	R/W	1
Executive Stack Pointer	ESP	01	R/W	1
Supervisor Stack Pointer	SSP	02	R/W	1
User Stack Pointer	USP	03	R/W	1
Interrupt Stack Pointer	ISP	04	R/W	1
P0 Base	P0BR	08	R/W	1
P0 Length	P0LR	09	R/W	1
P1 Base	P1BR	0A	R/W	1
P1 Length	P1LR	0B	R/W	1
System Base	SBR	0C	R/W	1
System Length	SLR	0D	R/W	2
Process Control Block Base	PCBB	10	R/W	1
System Control Block Base	SCBB	11	R/W	1
Interrupt Priority Level	IPL	12	R/W	2
Asynchronous System Trap Level	ASTL	13	R/W	2
Software Interrupt Request	SIRR	14	W	2
Software Interrupt Summary	SISR	15	R/W	2
Machine Check Status	MCSR	17	R	1
Interval Clock Control/Status	ICCS	18	R/W	1
Next Interval Count	NICR	19	R/W	1

Register Name	Acronym	Address	Type*	VAX** Processor
Interval Count	ICR	1A	W	1
Time-of-year	TODR	1B	R/W	1
Console Receive Control/Status	RXCS	20	R/W	1
Console Receive Data Buffer	RXDB	21	R/W	1
Console Transmit Control/Status	TXCS	22	R/W	1
Console Transmit Data Buffer	TXDB	23	R/W	1
Cache Error	CAER	27	R/W	1
Accelerator Control/Status	ACCS	28	R/W	1
Memory Management Enable	MAPEN	38	R/W	2
Translation Buffer Invalidate All	TBIA	39	W	2
Translation Buffer Invalidate Single	TBIS	3A	W	2
Performance Monitor Enable	PMR	3D	R/W	2
System Identification	SID	3E	R	1
Translation Buffer Check	TBCHK	3F	W	1

* R/W is read or write, R is read-only, and W is write-only

** (1) All VAX processor types
(2) All VAX processor types except VAX-11/725 and VAX-11/730

▪ SYSTEM IDENTIFICATION REGISTER

The system identification (SID) register is a read-only constant register that specifies the VAX processor type and the hardware and software revision levels contained in the processor. The values in the SID register are included in the error log. The type field may be used by software to distinguish processor types. Figure 10-1 shows the information format in the SID register.

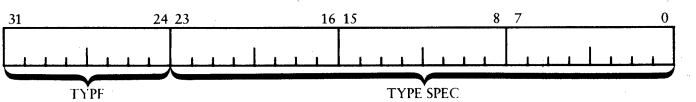


Figure 10-1 ▪ System Identification Register Format

Bit	Function
31:24	TYPE–A binary number that identifies the model of the VAX processor as follows: **Type** **Assignment** 0 Reserved for Digital Equipment Corporation

Bit	Function	
	1	VAX-11/780 or VAX-11/785 processor as specified by bits 23:22
	2	VAX-11/750 processor
	3	VAX-11/730 processor
	4	VAX 8600 processor
	5-127	Reserved for Digital Equipment Corporation
	128-255	Reserved for Digital's Computer Special Systems (CSS) group and customers
23:0	TYPE SPEC (Type specific)–Binary numbers used to identify the microcode version and hardware revision levels assigned to the system. The following bits are assigned to the processor types.	
	VAX-11/725 and VAX-11/730 Processors: Bits 23:16–Not used Bits 15:8–The software revision level of the CPU microcode Bits 7:0–The revision level of the CPU hardware	
	VAX-11/750 and VAX-11/751 Processors: Bits 23:16–Not used Bits 15:8–The software revision level of the CPU microcode Bits 7:0–The revision level of the CPU hardware	
	VAX-11/780 and VAX-11/785 Processors: Bits 23:22–The type of VAX processor where 1 is the VAX-11/780 processor and 0 is the VAX-11/785 Processor Bits 21:18–The revision level of the CPU hardware Bits 17:15–The letter assigned to the system revision level Bits 14:12–The number assigned to the Digital Equipment Corporation facility where the CPU was manufactured Bits 11:0–The serial number assigned to the CPU	
	VAX-11/782 Processor: Bits 23:16–The engineering revision level of the CPU hardware Bits 15:12–The number assigned to the Digital Equipment Corporation facility where the CPU was manufactured Bits 11:0–The serial number assigned to the CPU	
	VAX 8600 Processor: The same configuration as the VAX-11/782 processor	

- PROCESS CONTROL BLOCK

The process control block, shown in figure 10-2, comprises locations in memory used to store register information that is required by a process. A process is the basic scheduled entity that is executed by the processor. The process consists of an address space and both hardware and software context. The hardware context of the process is defined by the process control block (PCB), which contains images of the general purpose registers, processor status longword (PSL), program counter (PC), per-process stack pointers, and process virtual memory, which is defined by the base and length registers. During the process execution, the contents of the PCB is moved into the internal registers, where the hardware context is continually updated. When no process is executing, the PCB stores the contents of the registers for future use.

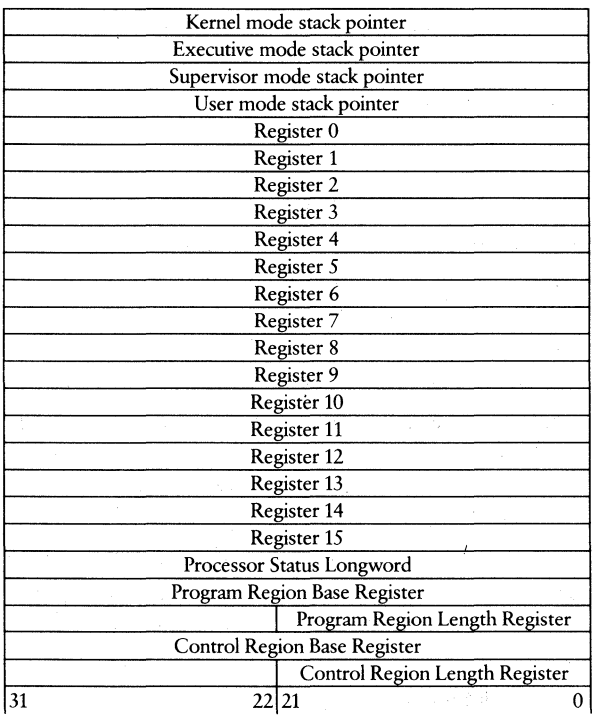


Figure 10-2 ▪ Process Control Block Structure

Kernel Stack Pointer Register The kernel stack pointer (KSP) is a read-write register that contains the address of the kernel-mode program used when the current access-mode field in the PSL is zero and the interrupt stack equals zero. Figure 10-3 shows the format of the KSP register.

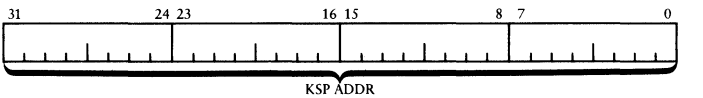


Figure 10-3 • Kernel Stack Pointer Register Format

Bit	Function
31:0	KSP ADDR (Kernel stack pointer address) – Contains the stack pointer address of the kernel-mode program.

Figure 10-4 • Executive Stack Pointer Register Format

Bit	Function
31:0	ESP ADDR (Executive stack pointer address) – Contains the stack pointer address of the executive-mode program.

Figure 10-5 • Supervisor Stack Pointer Register Format

Bit	Function
31:0	SSP ADDR (Supervisor stack pointer address) – Contains the stack pointer address of the supervisor-mode program.

Figure 10-6 • User Stack Pointer Register Format

Bit	Function
31:0	USP ADDR (User stack pointer address) – Contains the stack pointer address of the user-mode program.

Figure 10-7 • Interrupt Stack Pointer Register Format

Bit	Function
31:0	ISP ADDR (Interrupt stack pointer address) – Contains the last address of the process routine in memory prior to the interrupt.

Executive Stack Pointer Register The executive stack pointer (ESP) is a read-write register that contains the address of the executive-mode program selected when the current access-mode field in the PSL register is equal to zero. Figure 10-4 shows the format of the ESP register.

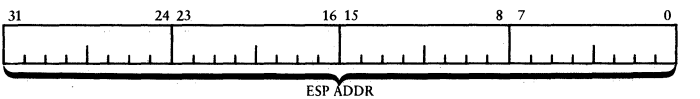


Figure 10-4 ▪ Executive Stack Pointer Register Format

Supervisor Stack Pointer Register The supervisor stack pointer (SSP) is a read-write register that contains the address of the supervisor-mode program selected when the current access mode field in the PSL register equals two. Figure 10-5 shows the format of the SSP register.

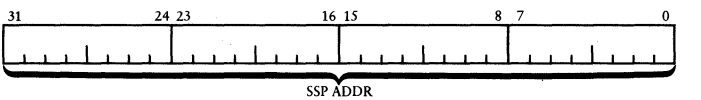


Figure 10-5 ▪ Supervisor Stack Pointer Register Format

User Stack Pointer Register The user stack pointer (USP) is a read-write register that contains the address of the user-mode program selected when the current access-mode field in the PSL equals three. Figure 10-6 shows the format of the USP register.

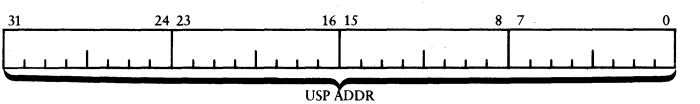


Figure 10-6 ▪ User Stack Pointer Register Format

Interrupt Stack Pointer Register When the processor changes states from the process context to the interrupt service context, the address information in general register R14 is loaded into the interrupt stack pointer (ISP)

register. This is a read-write register that insures a continuation of the process after the interrupt has been serviced. Figure 10-7 shows the format of the ISP.

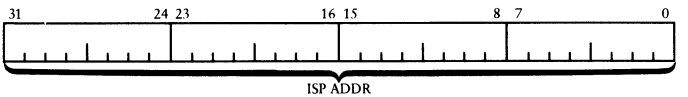


Figure 10-7 • Interrupt Stack Pointer Register Format

• VIRTUAL ADDRESS SPACE

The virtual address space is a linear array of 4 Gbytes of memory separated into 512-byte units called pages. The virtual address space has two parts, the per-process space and the system space, as shown in figure 10-8. The per-process space is the lower addressed half and is separated into the program region and the control region, each consisting of 1 Gbyte of memory. The system space consists of a system region and a reserved region, each containing 1 Gbyte of memory. The privileged registers used to select the memory locations are described in the following paragraphs.

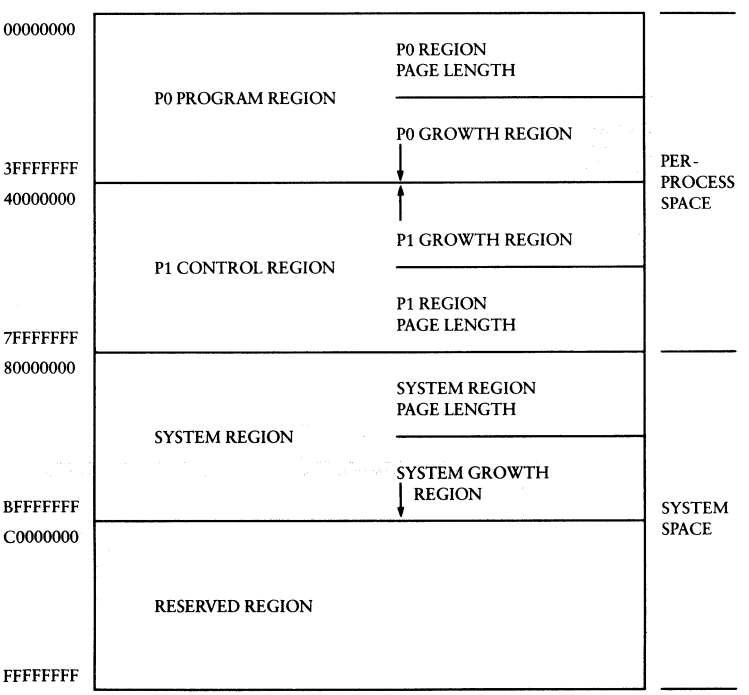


Figure 10-8 • Virtual Address Space Allocations

Per-Process Space Each process in the per-process space has a separate translation map; therefore, processes may not be contiguous. The program and control regions are defined by the following registers.

P0 Base Register The P0 program region of the virtual address space is mapped by the P0 page table (P0PT), which is defined by the P0 base register (P0BR) and by the P0 length register (P0LR). The P0BR is a read-write register containing the virtual address of the program region that is the base address of the P0PT. Figure 10-9 shows the format of the P0BR.

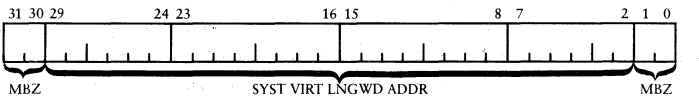


Figure 10-9 ▪ P0 Base Register Format

Bit	Function
31:30	MBZ (Must be zero).
29:2	SYST VIRT LNGWD ADDR (System virtual longword address) – Contains the base address of the P0 program region in virtual memory.
1:0	MBZ (Must be zero).

P0 Length Register The P0 length register (P0LR) is a read-write register that specifies the number of longwords in the P0PT. Figure 10-10 shows the format of the P0LR.

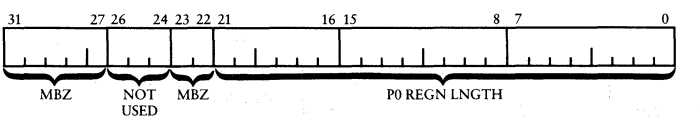


Figure 10-10 ▪ P0 Length Register Format

Bit	Function
31:27	MBZ (Must be zero).
26:24	Not used.
23:22	MBZ (Must be zero).
21:0	P0 REGN LNGTH (P0 region length) – The length of the P0 region in longwords.

P1 Base Register The P1 region of the address space is mapped by the P1 page table (P1PT), which is defined by the P1 base register (P1BR) and the P1 length register (P1LR). The P1BR is the read-write base register for the page table that describes the process virtual address. Figure 10-11 shows the format for the P1BR.

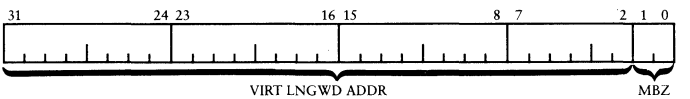


Figure 10-11 • P1 Base Register Format

Bit	Function
31:2	VIRT LNGWD ADDR (Virtual longword address) – The base address of the P1 control region in memory.
1:0	MBZ (Must be zero).

P1 Length Register The P1 length register (P1LR) is a read-write register that contains the length of the P1 control region in pages. The P1 space expands toward the smaller addresses; therefore, the P1LR contains the number of nonexistent page table entries (PTEs). Figure 10-12 shows the format of the P1LR.

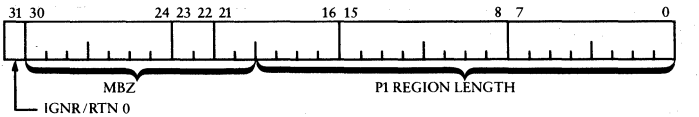


Figure 10-12 ▪ P1 Length Register Format

Bit	Function
31	IGNR/RTN 0 (Ignore/return as zero) – Ignored on a write instruction and returned as zero on a read instruction to the register.
30:22	MBZ (Must be zero).
21;0	P1 REGN LNGTH (P1 region length) – A binary number that is the length of the P1 region in longwords.

▪ SYSTEM SPACE

The system virtual address space is defined by the system page table (SPT), which is a vector of page table entries (PTEs). The SPT is located in the physical address space. The base address of the SPT is also a physical address and is contained in the system base register (SBR). The size of the SPT in longwords is contained in the system length register (SLR).

System Base Register The system base register (SBR) is a read-write register that contains the base address of the SPT and points to the first PTE in the table. The PTEs contain the mapping information or point to mapping information in the global page table. Figure 10-13 shows the format of the SBR.

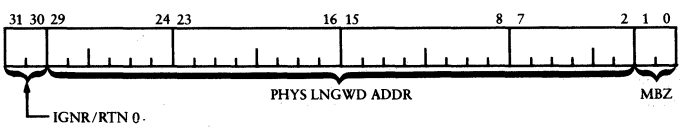


Figure 10-13 ▪ System Base Register Format

Bit	Function
31:30	MBZ (Must be zero).
29:2	PHYS LNGWD ADDR (Physical longword address)—Contains the base address of the SPT in memory.
1:0	MBZ (Must be zero).

System Length Register The system length register (SLR) is a read-write register that contains the length of the system region in pages which is the length of the SPT in longwords. Figure 10-14 shows the format of the SLR.

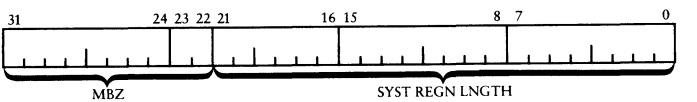


Figure 10-14 • System Length Register Format

Bit	Function
31:22	MBZ (Must be zero).
21:0	SYST REGN LNGTH (System region length)—Contains a binary number that specifies the length of the system region in longwords.

Process Control Block Base Register The process control block base (PCBB) is a read-write register that contains the physical longword address of the current executing process in the PCB. Figure 10-15 shows the format of the information in the register.

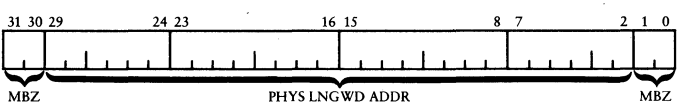


Figure 10-15 • Process Control Block Base Register Format

Bit	Function
31:30	MBZ (Must be zero).
29:2	PHYS LNGWD ADDR (Physical longword address)—Contains the physical longword address of the PCB.
1:0	MBZ (Must be zero).

System Control Block Base Register The system control block (SCB) is a page containing the vector addresses by which the exceptions and interrupts are dispatched to the appropriate service routines. The system control block base (SCBB) is a read-write register that contains the physical address of the SCB, as shown on figure 10-16.

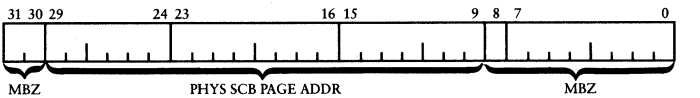


Figure 10-16 ▪ System Control Block Base Register Format

Bit	Function
31:30	MBZ (Must be zero).
29:9	PHYS SCB PAGE ADDR (Physical SCB page address)—A binary number that specifies the physical address of the system control block.
8:0	MBZ (Must be zero).

▪ PROGRAM INTERRUPTS

The processor services interrupt requests between instruction processing and at specific points during the execution of long interactive instructions. Fifteen priority-interrupt levels are assigned to software and all levels must be lower than the current processor-interrupt priority level.

Interrupt Priority Level Register The interrupt priority level (IPL) is a read-write register that enables the processor priority to be monitored or changed. During write operations to the IPL, the priority level from 19 through 1D (hexadecimal) can be selected and will be loaded into the processor status longword (PSL) bits 20 through 16. During read operations, the current priority level from the PSL is read from the IPL. Figure 10-17 shows the format of the IPL register.

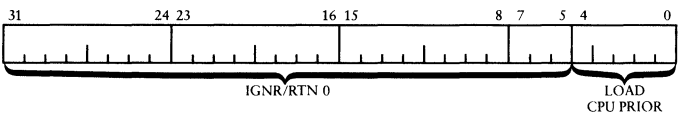


Figure 10-17 • Interrupt Priority Level Register Format

Bit	Function
31:5	IGNR/RTN 0 (Ignore/return as zero) – This field is ignored on a write instruction and returned as all zeros on a read instruction to the register.
4:0	LOAD CPU PRIOR (Load CPU priority) – A hexadecimal number from 19 to 1D (hexadecimal) that selects the interrupt priority level to be assigned to the processor level.

Asynchronous System Traps Register An asynchronous system trap (AST) is a software-simulated interrupt used to notify a process of events that are not synchronized with its execution and to initiate the processing of AST routines that service the event. A program can request an AST routine after an I/O operation is completed, when the time interval expires, when power is restored after a failure, and after other events. The asynchronous system trap level (ASTL) is a read-write register (ASTLR) containing the number of the most privileged access mode for which the AST is pending, as shown in figure 10-18.

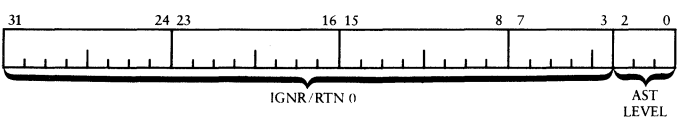


Figure 10-18 • Asynchronous System Trap Level Register Format

Bit	Function
31:3	IGNR/RTN 0 (Ignore/return as zero) – This field is ignored on a write instruction and returned as all zeros on a read operation.
2:0	AST LEVEL (Asynchronous system trap level) – The binary number that indicates the AST level of the access mode pending are as follows: **Value** **Mode** 0 Kernel 1 Executive 2 Supervisor 3 User 4 No AST pending 5-7 Reserved for Digital Equipment Corporation

Software Interrupt Request Register The software interrupt request register (SIRR) is a write-only register used for making software interrupt requests. One of 15 interrupt request levels can be specified during a write operation to register bits 3 through 0. The format of the SIRR is shown in figure 10-19.

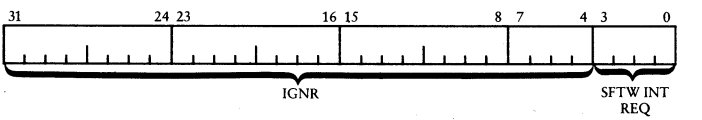


Figure 10-19 • Software Interrupt Request Register Format

Bit	Function
31:4	IGNR (Ignored) – These bits are ignored.
3:0	SFTW INTR REQ (Software interrupt request) – A hexadecimal number from 01 to 0F used to select a software interrupt level.

Software Interrupt Summary Level Register The software interrupts levels that are pending are recorded in the software interrupt summary register (SISR). The format of this read-write register is shown in figure 10-20.

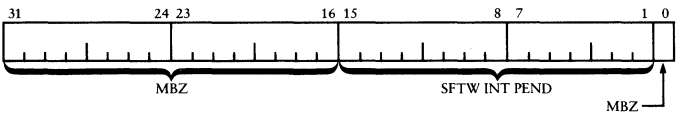


Figure 10-20 • Software Interrupt Summary Register Format

Bit	Function
31:16	MBZ (Must be zero).
15:1	SFTW INT PEND (Software interrupts pending)—Fifteen bit positions, 1 through F (hexadecimal), with each position set to indicate the level of the pending software interrupt and 0 as the lowest interrupt priority.
0	MBZ (Must be zero).

• SYSTEM CLOCKS

The VAX systems contain two clocks: a time-of-year clock and an interval clock. The time-of-year clock is used by the operating system to measure the duration of a power failure, to restart the system after a power loss, and to reinitialize the time and date after power has been restored. The time-of-year clock also is used to time-stamp user and operating system programs. The interval clock is used for accounting, for time-dependent events, and to maintain the software date and time.

Time-of-Year Register The time-of-year function is implemented by a read-write longword register (TODR) that forms an unsigned 32-bit binary counter and is driven by a precision clock source. The clock is accurate within 65 seconds per month (0.0025 percent); the range varies with ambient temperature and the charge on the battery backup unit. The battery backup will supply power for a maximum of 100 hours of operation after a power interruption and is recharged automatically. The clock count is continuous during transition to or from standby power. The least significant bit of the counter represents a resolution of 10 milliseconds. The counter cycles to zero after approximately 497 days. If the battery failure disrupts

the clock, the register is cleared on powerup and remains at zero until the time is reset by software. In the VAX-11/780 series processors, if the software initializes the clock to a value corresponding to a large unit of time such as a month, it can determine the length of downtime by checking the difference between the clock reading and the actual time that the power is restored. In the VAX-11/750 processor, the register remains cleared until a value is entered and the count then is initiated. The is TODR shown in figure 10-21.

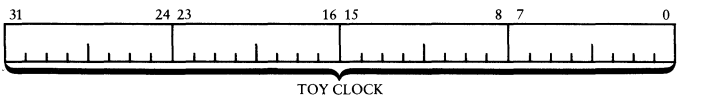


Figure 10-21 ▪ Time-of-Year Register Format

Bit	Function
31:0	TOY CLOCK (Time-of-year clock) – A 32-bit binary counter incremented at every 10-millisecond interval.

Interval Clock The interval clock provides an interrupt at IPL 24 at programmed intervals. The counter is incremented at one-microsecond intervals, with a typical accuracy of 0.01 percent or 8.64 seconds per day. The accuracy will vary slightly with ambient temperature changes. The clock interface consists of three registers: the interval count register (ICR), the next interval count register (NICR), and the interval clock control/status (ICCS) register.

Interval Count Register The interval count register (ICR) is a read-only register that is incremented once every microsecond. It is automatically loaded from the NICR when a carry bit (bit-31 overflow) occurs. A carry bit also causes an interrupt request at IPL 24 if the interrupt is enabled. Figure 10-22 shows the ICR format.

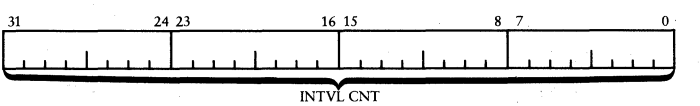


Figure 10-22 ▪ Interval Count Register Format

Bit	Function
31:0	INTVL CNT (Interval count)—Contains the interval count, which is incremented at one millisecond intervals.

Next Interval Count Register The next interval count register (NICR) is a write-only register that holds the value to be loaded into the ICR when the ICR overflows. The value is retained when the ICR is loaded. The NICR can be loaded regardless of the values contained in the ICR and ICCS. Figure 10-23 shows the format of the NICR.

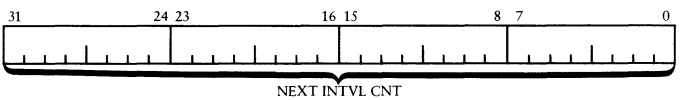


Figure 10-23 • Next Interval Count Register Format

Bit	Function
31:0	NEXT INTVL CNT (Next interval count)—Contains the next interval count value loaded by software.

Interval Clock Control/Status Register The interval clock control/status (ICCS) is a read-write register that contains control and status information for the interval clock. The format of the information in the register is shown in figure 10-24.

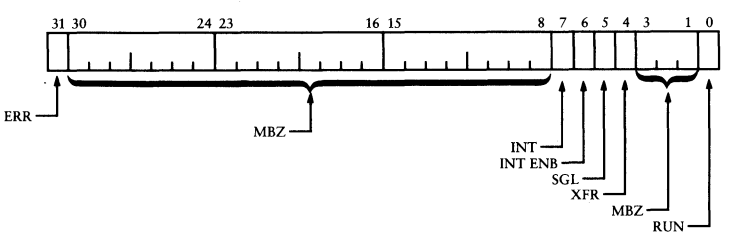


Figure 10-24 • Interval Clock Control/Status Register Format

Bit	Function
31	ERR (Error) – Set to indicate that one or more clock counts have been missed. This bit is set if INT (bit 7) is set when an ICR overflow occurs. An attempt to set this bit by the MTPR instruction will clear ERR.
30:8	MBZ (Must be zero).
7	INT (Interrupt) – Set by hardware when the ICR overflows. If the INT ENB (bit 6) is set, an interrupt is generated. An attempt to set this bit by the MTPR instruction will clear INT and reenable the clock count interrupt if INT ENB (bit 6) is set.
6	INT ENB (Interrupt enable) – When this bit is set, an interrupt request at IPL 24 is generated every time the ICR overflows. When this bit is clear, no interrupt is requested. If INT is already set and the software sets this bit, an interrupt is generated. This bit is cleared during the bootstrap loading sequence.
5	SGL (Single) – This is a write-only bit. If the RUN (bit 0) is clear when this bit is set, the ICR is incremented by one.
4	XFR (Transferred) – This is a write-only bit. Each time this bit is set, the value in the NICR is transferred to the ICR.
3:1	MBZ (Must be zero).
0	RUN – When set, the ICR is incremented every microsecond. When this bit is clear, the ICR does not increment automatically. At bootstrap loading time, this bit is cleared.

▪ CONSOLE TERMINAL REGISTERS

The console terminal is accessed through four internal registers. Two registers, the console receive control/status (RXCS) and console receive data buffer (RXDB), are used for receiving information from the console. The console transmit control/status (TXCS) register and the console transmit data buffer (TXDB) register are used for writing information to the console terminal. The functions and type of console registers vary, depending on the VAX system. Refer to the register description of each VAX processor for details on console register functions.

Console Receive Control/Status Register The console receive control/status (RXCS) is a read-write register that is cleared during hardware initialization. Figure 10-25 shows the format of the RXCS and describes the bit assignments.

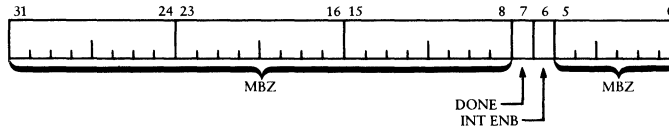


Figure 10-25 • Console Receive Control/Status Register Format

Bit	Function
31:8	MBZ (Must be zero).
7	DONE – A read-only bit that is set by the console when data is received by the RXDB and cleared when the MFPR instruction is performed with RXDB as the destination.
6	INT ENB (Interrupt enable) – When set by software, an interrupt is generated on the interrupt priority line (IPL 20) when DONE (bit 7) is set. If the DONE bit has been previously set and the software sets this bit, an interrupt is also generated.
5:0	MBZ (Must be zero).

Console Receive Data Buffer Register The console receive data buffer (RXDB) is a read-only register that receives information from the terminal for transmission to the processor. Figure 10-26 shows the format of the RXDB and the bit assignments.

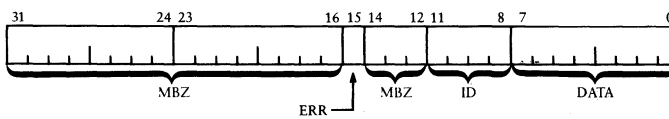


Figure 10-26 • Console Receive Data Buffer Register Format

Bit	Function
31:16	MBZ (Must be zero).
15	ERR (Error) – Set if the received data contains an error such as overrun or loss of connection.
14:12	MBZ (Must be zero).
11:8	ID (Identification) – A hexadecimal number that identifies the source or function of the data (bits 7:0) received by the console subsystem. ID = 0: console terminal ID = 1 through F: implementation dependent
7:0	DATA – The data received from the console subsystem device.

Console Transmit Control/Status Register The console transmit control/status (TXCS) is a read-write register that contains the console transmit control and status information. Except for the RDY bit, the register is cleared during hardware initialization. Figure 10-27 shows the format of the TXCS register.

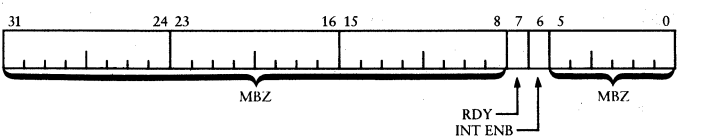


Figure 10-27 • Console Transmit Control/Status Register Format

Bit	Function
31:8	MBZ (Must be zero).
7	RDY (Ready) – A read-only bit set during bootstrap loading and when the console transmitter is not busy. Cleared when the MTPR instruction is executed with the TXDB register designated as the source.

Bit	Function
6	INT ENB (Interrupt enable) – A read-only bit set by software to generate an interrupt request priority line (IPL 20) when the RDY (bit 7) is set. If the RDY bit is already set and the INT ENB bit is set by software, an interrupt is also generated. This bit is cleared when an MTPR instruction is executed with the TXDB register designated as the source.
5:0	MBZ (Must be zero).

Console Transmit Data Buffer Register The console transmit data buffer (TXDB) is a write-only register that receives the data from the processor for transmission to the console terminal. Figure 10-28 shows the format of the TXDB register and describes the bit assignments.

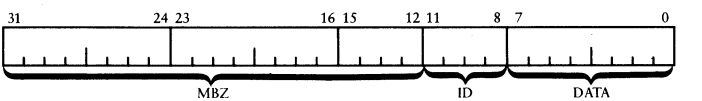


Figure 10-28 • Console Transmit Data Buffer Register Format

Bit	Function
31:12	MBZ (Must be zero).
11:8	ID (Identification) – A hexadecimal value used as a select code to identify the console terminal, console storage device, or communication function, as follows:

Select code (ID)	Device	Data Field
0	Console terminal	ASCII data (0 through 7F hexadecimal)
1	Console drive 0	Binary data (0 through FF hexadecimal)
2	Function complete complete	Status
5*	Protocol error	Ignored

Bit	Function		
	9	Drive 0 command command	0 = read sector 1 = write sector 2 = read status 3 = write deleted data sector 4 = cancel function 5 = protocol error error
	F	Miscellaneous communication	0 = no operation 1 = software done 2 = boot CPU 3 = clear restart flag 4 = clear bootstrap flag
* Sent by the console when a load device command is issued before a previous command is completed or the console receives a select code 1 when expecting a command.			
7:0	DATA – The data, status, or function information of the device selected by the ID (bits 11:8).		

▪ MEMORY REGISTERS

Several memory registers are available to control the memory management function and the translation of physical addresses to virtual addresses.

Memory Management Enable Register The translation of a virtual address to a physical address is controlled by the memory management enable register also called the map enable register (MAPEN). This read-write register is cleared during the processor initialization. Figure 10-29 shows the format of the MAPEN register.

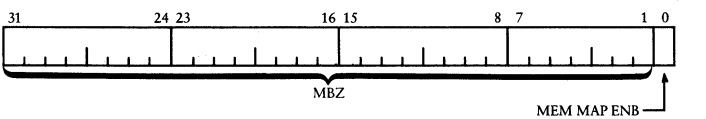


Figure 10-29 ▪ Memory Management Enable Register Format

Bit	Function
31:1	MBZ (Must be zero).
0	MEM MAP ENB (Memory map enable)—Set to enable the memory management control.

Translation Buffer Invalidate All Register The translation buffer invalidate all (TBIA) is a write-only register used to invalidate the process pages in the translation buffer when the software changes a system page-table entry. The format of the TBIA is shown in figure 10-30 and writes all zeros into the translation buffer.

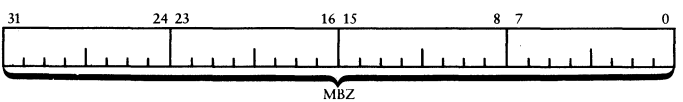


Figure 10-30 • Translation Buffer Invalidate All Register Format

Bit	Function
31:0	MBZ (Must be zero).

Translation Buffer Invalidate Single Register The translation buffer invalidate single (TBIS) is a write-only register used to move a virtual address into the translation buffer when the software updates any part of a valid page table entry for the system or a current process region. The format of the address information is shown in figure 10-31.

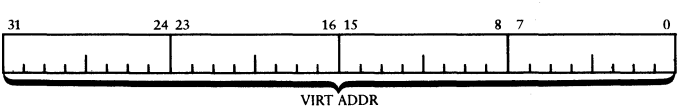


Figure 10-31 • Translation Buffer Invalidate Single Register Format

Bit	Function
31:0	VIRT ADDR (Virtual address) – The virtual address of the current memory reference.

Translation Buffer Check Register The translation buffer check (TBCHK) is a write-only register available for checking the presence of a valid translation in the translation buffer. Figure 10-32 shows the format of the information in the register.

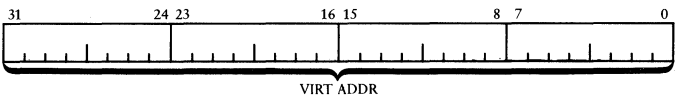


Figure 10-32 ▪ Translation Buffer Check Register Format

Bit	Function
31:0	VIRT ADDR (Virtual address) – The virtual address to be checked in the translation buffer. If a valid translation is performed for the virtual page, the condition code V bit is set.

Machine Check Status Register The machine check status register (MCSR) is a read-only register used with the machine check error status (MCESR) register to provide additional information about bus errors and translation buffer errors that caused a machine check. The register is cleared during the bootstrap loading sequence. The format of the information in the MCSR is shown in figure 10-33 and the bit assignments are described.

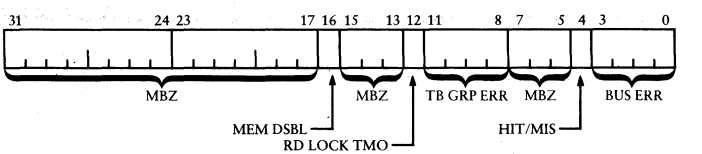


Figure 10-33 ▪ Machine Check Status Register Format

Bit	Function
31:17	MBZ (Must be zero).
16	MEM DSBL (Memory disable) – A read-only by software bit that is set to allows only the cache memory to be read from or written to and main memory cannot be referenced or modified. This results in a failure of the CPU.
15:13	MBZ (Must be zero).
12	RD LOCK TMO (Read lock timeout) – A software read-only bit that is set by a read lock timeout and defines the interpretation of the bus error (bits 3:0).
11:8	TB GRP ERR (Translation group errors) – Read-only bits that define the type of translation buffer group parity error and are cleared when TB error (bit 2) of the MCESR is reset.
7:5	MBZ (Must be zero).
4	HIT/MIS (Hit/Miss) – A read-only bit that is set when the microcoded reference results in a hit or a miss.
3:0	BUS ERR (Bus error) – Read-only bits that are set to define the type of bus error. They are also cleared when bus error (bit 3) of the MCESR is cleared. The function of the bits when set are as follows: Bit 3 – Indicates that the memory location addressed does not exist or that a read-lock timeout has occurred. Bit 2 – Indicates that an uncorrectable data error has occurred. Bit 1 – Indicate that a bus error has occurred and was not corrected. Bit 0 – Indicates that the data error has been corrected.

VAX-11/725 and VAX-11/730 Processor-specific Registers

In addition to the architectural processor registers, the VAX-11/725 and VAX-11/730 processors contain the specific registers defined in this section and listed in table 10-2. Table 10-2 lists the register acronym, the hexadecimal processor address, and the register type.

Table 10-2 ▪ VAX-11/725 and VAX-11/730 Processor-specific Registers

Register Name	Acronym	Address	Type*
Console Storage Receive/Status	CSRS	1C	R/W
Console Storage Receive Data	CSDR	1D	R/W
Console Storage Transmit/Status	CSTS	1E	R/W
Console Storage Transmit Data	CSTD	1F	R/W
Accelerator Control/Status	ACCS	28	R/W

* R/W is read or write

▪ TU58 CONSOLE STORAGE

The TU58 console tape-cartridge subsystem is accessed through four internal registers. The console storage receive status (CSRS) and the console storage receive data (CSRD) registers are associated with receiving information from the TU58 drive for transmission to the processor. The console storage transmit status (CSTS) and the console storage transmit data (CSTD) registers are associated with writing information on the tape from the processor.

Console Storage Receive Status Register The console storage receive status (CSRS) is a read-write register used to control receive operations and to transmit receive-status information to the processor. Figure 10-34 shows the format of the assigned bits in the register.

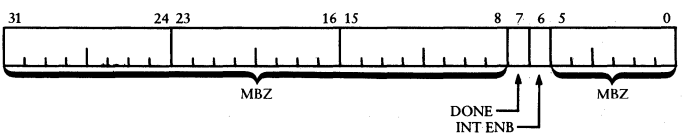


Figure 10-34 ▪ Console Storage Receive Status Register Format

Bit	Function
31:8	MBZ (Must be zero).
7	DONE – A read-only bit set by the TU58 tape drive whenever a byte of data is received by the TU58 from the CSRD register. It is cleared during hardware initialization whenever a MFPR instruction is executed with the CSRD register as the destination.
6	INT ENB (Interrupt Enable 0) – Set by software to enable an interrupt to be generated on the priority line (IPL 23). If the DONE bit already has been set and the IE bit is set by the software, an interrupt is generated. This bit is cleared during hardware initialization.
5:0	MBZ (Must be zero).

Console Storage Receive Data Register The console storage receive data (CSRD) is a read-only register that receives the information from the TU58 tape drive for transfer to the processor. Figure 10-35 shows the format of the CSRD register.

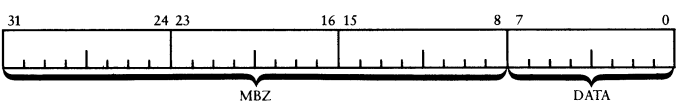


Figure 10-35 ▪ Console Storage Receive Data Register Format

Bit	Function
31:8	MBZ (Must Be zero).
7:0	DATA – This field contains a byte of data received from the TU58 tape subsystem.

Console Storage Transmit Status Register The console storage transmit status (CSTS) is a read-write register that provides the control and status for the TU58 transmit operations. The format for the CSTS register is shown in figure 10-36.

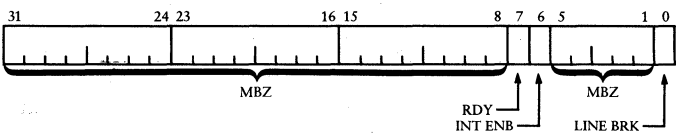


Figure 10-36 ▪ Console Storage Transmit Status Register Format

Bit	Function
31:8	MBZ (Must be zero).
7	RDY (Ready) – A read-only bit that is set during bootstrap loading, and when the TU58 tape drive is not busy.
6	INT ENB (Interrupt enable) – Set by software to generate an interrupt at IPL 23. If the RDY (bit 6) already has been set, an interrupt request will be generated when this bit is set.
5:1	MBZ (Must be zero).
0	LINE BRK (Line break) – A write-only bit set to generate a line break.

Console Storage Transmit Data Register The console storage transmit data (CSTD) is a write-only register that contains the data to be transferred to the TU58 drive from the processor. Figure 10-37 shows the format of the information in the CSTD register.

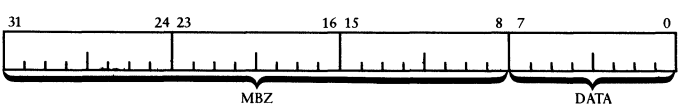


Figure 10-37 ▪ Console Storage Transmit Data Register Format

Bit	Function
31:8	MBZ (Must be zero).
7:0	DATA – Contains the data for transfer to the TU58 tape drive.

• FP730 FLOATING-POINT ACCELERATOR

The FP730 floating-point accelerator option contains an internal read-write accelerator control/status (ACCS) register that indicates whether an FP730 option is installed, controls whether it is enabled, identifies its type, and reports errors and status. During hardware initialization, the type and enable values are set and any error indications are cleared. Figure 10-38 shows the format of the information in the ACCS register.

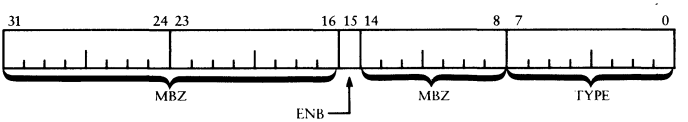


Figure 10-38 • FP730 Accelerator Control/Status Register Format

Bit	Function
31:16	MBZ (Must be zero).
15	ENB (Enable) – A read-only bit that specifies whether the accelerator is enabled. This bit is set if the accelerator is installed and functioning.
14:8	MBZ (Must be zero).
7:0	TYPE – A read-only binary number that specifies the accelerator type as follows:
	0 = No accelerator 1 = Floating-point accelerator 2 through 127 = Reserved for Digital Equipment Corporation 128 through 255 = Reserved to Digital's CSS group customers

VAX-11/750 and VAX-11/751 Processor-specific Registers

In addition to the architectural processor registers, the VAX-11/750 and VAX-11/751 processors contain the specific registers defined in this section and listed in table 10-3. Table 10-3 lists the register acronym, the hexadecimal processor address, and the register type.

Table 10-3 ▪ VAX-11/750 and VAX-11/751 Processor-specific Registers

Register Name	Acronym	Address	Type*
CMI Error	CMIER	17	R
Console Storage Receive/Status	CSRS	1C	R/W
Console Storage Receive Data	CSDR	1D	R/W
Console Storage Transmit/Status	CSTS	1E	R/W
Console Storage Transmit Data	CSTD	1F	R/W
Translation Buffer Disable	TBDR	24	R/W
Cache Disable	CADR	25	R/W
Machine Check Error Summary	MCESR	26	R/W
Accelerator Control/Status	ACCS	40	R/W
Initialize UNIBUS	IORESET	37	W
Translation Buffer Data	TBDATA	59	R/W
Accelerator Control/Status	ACCS	28	R/W

* R/W is read or write, W is write-only

- TU58 CONSOLE STORAGE

  The TU58 console tape-cartridge subsystem is accessed through four internal registers that are identical in format and function to the TU58 registers previously described for the VAX-11/725 and VAX-11/730 privileged processor registers.

- BUS REGISTERS

  The status of the CMI bus is monitored and the UNIBUS initialization is controlled by the processor registers described in the following paragraphs.

  CMI Error Register The computer memory interconnect error register (CMIER) is a read-only register that contains the bus error status information to enable correction of the indicated error. Figure 10-39 shows the format of the register and describes the function of the bits.

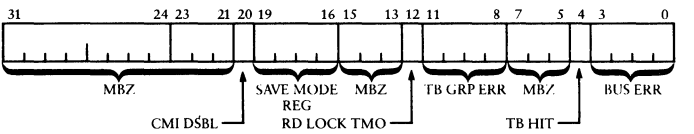


Figure 10-39 • CMI Error Register Format

Bit	Function
31:21	MBZ (Must be zero).
20	CMI DSBL (CMI disable) – Set to disable the CMI references.
19:16	SAVE MODE REG (Saved mode register) – Indicates the processor access mode for the last memory reference as follows:
	Bit 19 – not used Bit 18 – 0 is virtual memory, 1 is physical memory Bits 17:16 – processor access mode 1 through 4
15:13	MBZ (Must be zero).
12	RD LOCK TMO (Read Lock Timeout) – Set to indicate that a read-lock timeout has occurred.
11:8	TB GRP ERR (Translation buffer group error) – These bits are set to indicate a data or tag error, as follows: Bit 8 – TB group 0 data error Bit 9 – TB group 1 data error Bit 10 – TB group 0 tag error Bit 11 – TB group 1 tag error
7:5	MBZ (Must be zero).
4	TB HIT (Translation buffer hit) – Set to indicate that a translation buffer hit occurred on the last reference.
3:0	BUS ERR (Bus error) – Set to indicate that the action performed on the error condition is as follows: Bit 0 – corrected data error Bit 1 – lost error Bit 2 – uncorrectable data error Bit 3 – nonexistent memory

Initialize UNIBUS Register The initialize UNIBUS register (IUR) is a write-only register used for the software initialization of the UNIBUS. The format of the IUR information is shown in figure 10-40.

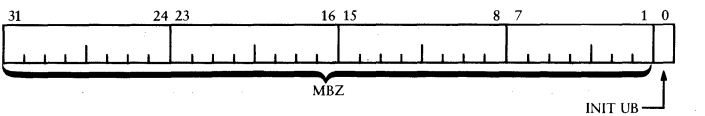


Figure 10-40 ▪ Initialize UNIBUS Register Format

Bit	Function
31:1	MBZ (Must be zero).
0	INIT UB (Initialize UNIBUS) – Set to initialize the UNIBUS memory and registers.

▪ MEMORY REGISTERS

Several memory registers are available to control the memory management function and the translation of physical addresses to virtual addresses.

Translation Buffer Disable Register The translation buffer disable register (TBDR) is a read-write register that controls the enabling and disabling of the translation buffer and determines which half of the buffer is replaced. The buffer is cleared during the bootstrap loading sequence. The format for the TBDR is shown in figure 10-41 and the bit assignments are described.

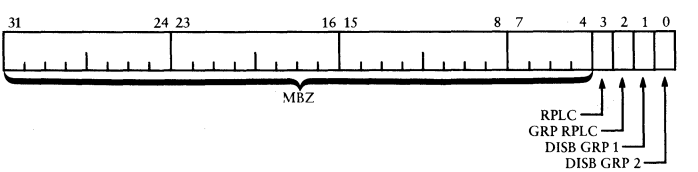


Figure 10-41 ▪ Translation Buffer Disable Register Format

Bit	Function
31:4	MBZ (must be zero).
3	RPLC (Replace) – When set, the GRP RPLC (bit 2) selects the half of the translation buffer that is replaced.
2	GRP RPLC (Group replace) – When the RPLC (bit 3) is set and this bit is cleared, group zero will be replaced. When bit 3 is set and this bit is also set, group one will be replaced.
1	DISB GRP 1 (Disable group 1) – Set to disable group one of the translation buffer.
0	DISB GRP 0 (Disable group 0) – Set to disable group zero of the translation buffer.

Translation Buffer Data Register The translation buffer data (TBDA) is a read-write register used to read data and write data to locations in the translation buffer. This capability is important for diagnostic purposes. The MTPR and MFPR instructions are two-operand instructions. One operand specifies a longword source or destination and the other operand specifies the IPR number of the TBDA. A third operand is required for accessing the translation buffer. It is the virtual address of the translation buffer location to be referenced. For MTPR instructions, the P0BR contains the virtual address whose PTE (taken from the source operand) is to be written into the translation buffer. For MFPR instructions, the P0BR contains the virtual address whose PTE is to be read from the translation buffer into the destination. Memory management must be disabled to perform these operations. Figure 10-42 shows the format of the information contained in the TBDA.

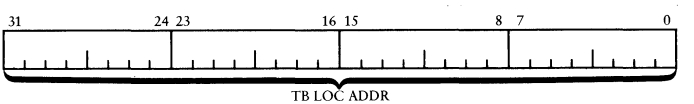


Figure 10-42 ▪ Translation Buffer Data Register Format

Bit	Function
31-0	TB LOC ADDR (Translation buffer location address) – Contains the address of the data to be read or written into the translation buffer.

Cache Disable Register The cache disable register (CADR) is a read-write register used to disable the cache memory. The CACH DSABL (bit 1) is cleared on powerup. Figure 10-43 shows the format of the information in the CADR.

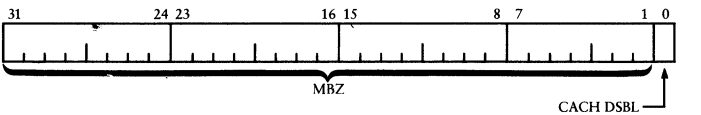


Figure 10-43 • Cache Disable Register Format

Bit	Function
31:1	MBZ (Must be zero).
0	CACH DSABL (Cache disable) – A read-write bit that is set to disable the cache memory.

Cache Error Register The cache error register (CAER) is a read-write register that logs information related to cache parity errors. All bits are cleared during the bootstrap loading sequence. This register format is shown in figure 10-44 and the bit assignments are described.

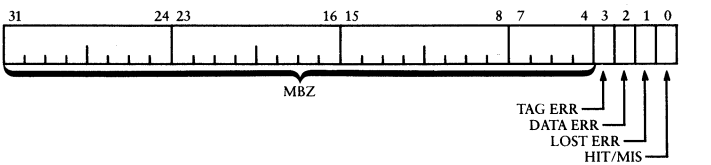


Figure 10-44 • Cache Error Register Format

Bit	Function
31:4	MBZ (Must be zero).
3	TAG ERR (Tag error) – Set to indicate that the parity error was in the tag portion of the cache.
2	DATA ERR (Data error) – Set to indicate that the parity error was in the data portion of the cache.
1	LOST ERR (Lost error) – Set to indicate that there were multiple cache errors.
0	HT/MIS (Hit/Miss) – Cleared to indicate that the last microcode reference resulted in a miss. Set to indicate that the last microcoded reference resulted in a hit.

Machine Check Error Summary Register The machine check error summary register (MCESR) is a read-write register that logs information pertaining to the causes of a machine check, such as a bus error. The register is cleared during the bootstrap loading sequence. The register format is shown in figure 10-45.

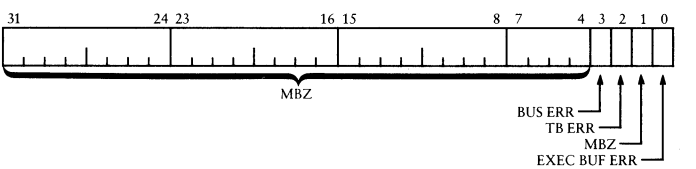


Figure 10-45 ▪ Machine Check Error Summary Register Format

Bit	Function
31:4	MBZ (Must be zero).
3	BUS ERR (Bus error) – Set when a machine check occurs as a result of either a read operation to a nonexistent memory location or when an error in the data read is uncorrectable.

Bit	Function
2	TB ERR (Translation buffer error) – Set when a machine check occurs as a result of a translation buffer error. It is cleared by writing a one to this bit position.
1	MBZ (Must be zero).
0	EXEC BUF ERR (Execution buffer error) – Set when an error is detected while trying to use data from the execution buffer. It is cleared by writing a one to this bit position.

FP750 Floating-point Accelerator The FP750 floating-point accelerator option includes an accelerator control/status (ACCS) register used to control and monitor the FPA operations. The functions and register format of the FP750 are the same as those previously described for the FP730 floating-point accelerator.

VAX-11/780, VAX-11/782, and VAX-11/785 Processor-specific Registers

In addition to the architectural processor registers, the VAX-11/780, VAX-11/782, and VAX-11/785 processors contain the specific registers defined in this section and listed in table 10-4. Table 10-4 lists the register acronym, the hexadecimal processor address, and the register type.

Table 10-4 • VAX-11/780, VAX-11/782, and VAX-11/785 Processor-specific Registers

Register Name	Acronym	Address	Type*
Accelerator Control/Status	ACCS	28	R/W
Accelerator Maintenance	ACCR	29	R/W
Writable Control Store Address	WCSA	2C	R/W
Writable Control Store Data	WCSD	2D	R/W
SBI Fault/Status	SBIFS	30	R/W
SBI Silo Data	SBIS	31	R
SBI Silo Comparitor	SBISC	32	R/W
SBI Maintenance	SBIMT	33	R/W
SBI Error	SBIER	34	R/W
SBI Timeout Address	SBITA	35	R
SBI Quadword Clear	SBIQC	36	W
Microprogram Breakpoint	MBRK	3C	R/W

* R/W is read or write, R is read-only, and W is write-only.

- MICRO CONTROL STORE

Three registers are provided for control and status of the processor microcode: the writable control store address (WCSA) register, the writable control store data (WCSD) register, and the microprogram breakpoint address (MBRK) register.

Writable Control Store Address Register The writable control store address (WCSA) is read-write address register that contains the address of the writable control store (WCS) locations to be read or written, a counter, and a parity indicator. Figure 10-46 shows the format of the WCSA information.

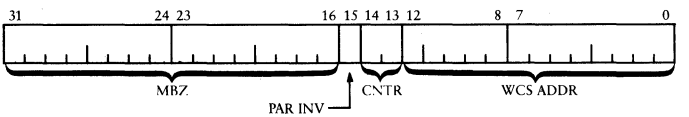


Figure 10-46 • Writable Control Store Address Register Format

Bit	Function
31:16	MBZ (Must be zero).
15	PAR INV (Parity inverted)—When set, the data written to the WCS includes an inverted parity indicator.
14:13	CNTR (counter)—A binary counter that is incremented as each 32-bit word is written and is cleared at the count of three as the WCS address is incremented.
12:0	WCS ADDR (Writable control store address)—Contains the address of the WCS location being written into.

Writable Control Store Data Register The WCSD is a read-write register that contains the data to be written or indicates the control store addresses that are available for user microcode. Figure 10-47 shows the format of the information in the WCSD.

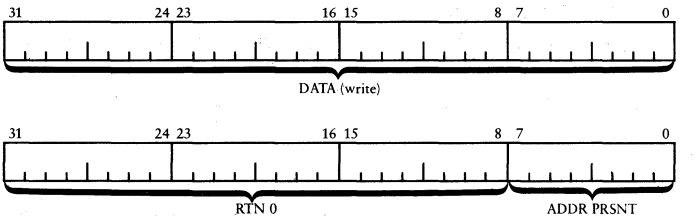


Figure 10-47 ▪ Writable Control Store Data Register Format

Bit	Function
31:0	DATA – Contains the data to be written into the WCS location during a write operation.
31:8	RTN 0 (Return as zero) – During a read operation, these bits are all zeros.
7:0	ADDR PRSNT (Addresses present) – During a read operation, the address is a binary number that indicates the addresses presently being used.

The WCSD register is read to indicate which control store addresses are available for writing. If the addresses present (bits 7:0) of the WCSD contain a number (n), then addresses n to the power of 1024 through zn to the power of 1024 + 1023 are available for writing microcode. The field defined by the number 4 is reserved for Digital Equipment Corporation. The remaining numbers define fields where blocks of microcode can be written by customers or by Digital. Each word of control store contains 96 bits plus 3 parity bits. An MTPR instruction to the WCSD register will write 32 bits and increment the counter (bits 14:13) of the WCSA. When the count is three, it is automatically cleared and the writable control store address is incremented. When a microword with bad parity is executed, a machine check will result. During bootstrap loading, the contents of the WCSA register are unpredictable.

Microprogram Breakpoint Address Register The microprogram breakpoint address (MBRK) is a read-write register that contains a breakpoint address. Whenever the microprogram PC address is the same as the contents of the MBRK register, a signal is asserted. If the console has enabled a stop on microbreak, then the processor clock is stopped when this signal is asserted. This signal is available as a diagnostic test point. During the bootstrap loading, the contents of the MBRK register are unpredictable. Figure 10-48 shows the format of the MBRK register information.

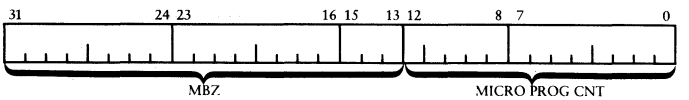


Figure 10-48 • Microprogram Breakpoint Address Register Format

Bit	Function
31:13	MBZ (Must be zero).
12:0	MICRO PROG CNT (Microprogram count) – A binary address that causes a microprogram break when the address is the same as the address in the microprogram counter.

• SYNCHRONOUS BACKPLANE INTERCONNECT BUS

All communication with I/O devices is through the synchronous backplane interconnect (SBI) bus and adapter. Several registers are provided to monitor and control the SBI functions during normal operations and during maintenance. The function of the registers are described in the following paragraphs.

SBI Fault/Status Register The SBI fault/status (SBIFS) is a read-write register that provides fault and status indications of the SBI adapter. Figure 10-49 shows the format of the information in the register.

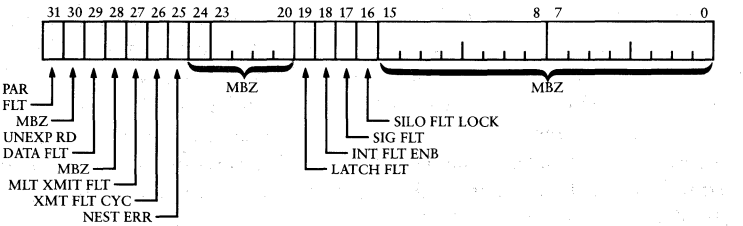


Figure 10-49 ▪ SBI Fault/Status Register Format

Bit	Function
31	PAR FLT (Parity fault) – A read-only bit set to indicate that a parity fault has been detected.
30	MBZ (Must be zero).
29	UNEXP RD DATA FLT (Unexpected read data fault) – A read-only bit set to indicate that a read data fault has been detected.
28	MBZ (Must be zero).
27	MLT XMIT FLT (Multiple transmitter fault) – A read-only bit set to indicate that a multiple transmitter fault has been detected.
26	XMIT FLT CYC (Transmitter during fault cycle) – A read-only bit set to indicate that a transmitter attempted to send data during a fault condition.
25	NEST ERR (Nested error).
24:20	MBZ (Must be zero).
19	LATCH FLT (Fault latch) – A write control bit set to latch a fault condition.
18	INT FLT ENB (Fault interrupt enable) – Set to enable a program interrupt on a fault condition.
17	SIG FLT (Fault signal) – A read-only bit set to indicate that the fault signal has occurred.
16	SILO FLT LOCK (Fault silo lock) – A write control bit set if the silo register is locked because of a fault condition.
15:0	MBZ (Must be zero).

SBI Silo Data Register The SBI silo data (SBIS) is a read-only register that records the state of the indicated SBI signals for the preceding 16 bus cycles. The silo is updated every cycle until the fault line of the SBI is asserted or an SBI silo comparator match occurs. Figure 10-50 shows the format of the silo data register.

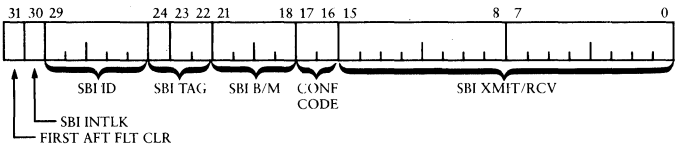


Figure 10-50 • SBI Silo Data Register Format

Bit	Function
31	FIRST AFT FLT CLR (First entry after fault cleared)—Set to indicate that the silo has received the first entry after the fault condition has cleared.
30	SBI INTLK (SBI interlock)—Set to indicate that the silo is interlocked.
29:25	SBI ID (SBI identifier)—A binary number used to identify the logical source of data during a write command or the logical destination of the data during a read command.
24:22	SBI TAG—A binary number that indicates the type of transmitted data on the information lines as follows:

Bit 24	Bit 23	Bit 22	Function
0	0	0	Read data
0	1	1	Command address
1	0	1	Write data
1	1	0	Interrupt summary read

21:18	SBI B/M (SBI M3:M0 or B31:B28)—Contains information field bits B31:B28 or mask field bits M3:M0 when the tag field specifies a command address tag.

Bit	Function
17:16	CONF CODE (Confirmation code) – A code that indicates the status of the confirmation lines in response to information transfers as follows: **Code** **Definition** 00 No response to a commander's selection 01 A positive acknowledgement to a transfer 10 The response to a command/address transfer indicating the succesful selection of a nexus that is presently unable to execute the command 11 The response to a command /address transfer that indicates the selection of a nexus that cannot execute the commamd
15:0	SBI XMIT/RCV (SBI transmit/receive) – Contains the information contained on the address/data lines during receive and transmit operations.

SBI Silo Comparitor Register The SBI silo comparator (SBISC) is a read-write register that allows the SBI information to become locked when specified conditions are detected. Both conditional and unconditional lock modes are provided. Figure 10-51 shows the format of the information in the register.

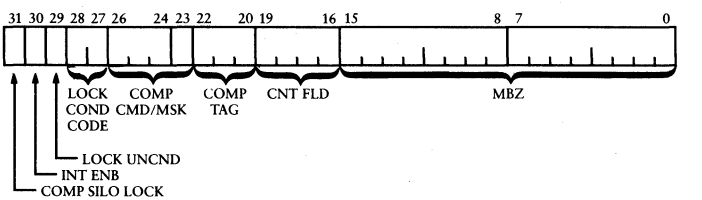


Figure 10-51 ▪ SBI Silo Comparator Register Format

Bit	Function
31	COMP SILO LOCK (Compare silo lock) – Set to indicate that a silo lock has occurred; cleared on a write to the SBISC register.
30	INT ENB (Interrupt enable) – Set to enable a program interrupt when the silo is locked.
29	LOCK UNCND (Lock unconditional) – Set to cause an unconditional lock of the silo information.
28:27	LOCK COND CODE (Conditional lock codes).
26:23	COMP CMD/MSK (Compare command/mask).
22:20	COMP TAG (Compare tag).
19:16	CNT FLD (Count field).
15:0	MBZ (Must be zero).

SBI Maintenance Register The SBI maintenance (SBIMT) is a read-write register that allows error conditions to be simulated for diagnostic maintenance functions. Figure 10-52 shows the format of the information in the SBIMT register.

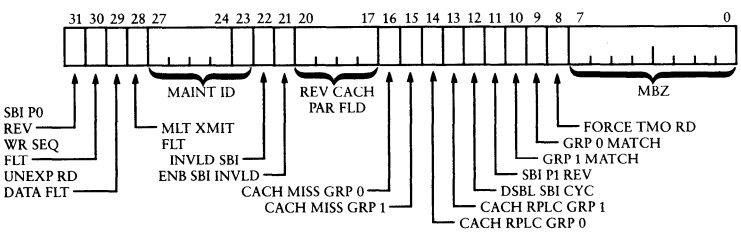


Figure 10-52 • SBI Maintenance Register Format

Bit	Function
31	SBI P0 REV (SBI P0 reversal) – Set to cause a reversal in the parity of the SBI P0 parity bit.
30	WR SEQ FLT (Write sequence fault) – Set to cause a write sequence fault to occur.
29	UNEXP RD DATA FLT (Unexpected read data fault) – Set to cause an unexpected read data fault.
28	MLT XMIT FLT (Multiple transmitter fault).
27:23	MAINT ID (Maintenance ID) – Set to a binary number to cause faults and used as a silo comparator.
22	INVLD SBI (Invalidate SBI) – Set to cause an SBI write invalidate to cache.
21	ENB SBI INVLD (Enable SBI invalidate) – Set to enable an SBI invalidate.
20:17	REV CACH PAR FLD (Reverse cache parity field) – Set to reverse the parity field of the cache memory.
16	CACH MISS GRP 0 (Cache miss group 0) – Set to cause a cache miss group 0.
15	CACH MISS GRP 1 (Cache miss group 1) – Set to cause a cache miss group 1.
14	CACH RPLC GRP 0 (Cache replacement group 0) – Set to cause a cache replacement group 0.
13	CACH RPLC GRP 1 (Cache replacement group 1) – Set to cause a cache replacement group 1.
12	DSBL SBI CYC (Disable SBI cycles) – Set to disable the cycles of the SBI bus.
11	SBI P1 REV (SBI PI reversal) – Set to cause a reversal in the parity of the SBI P1 parity bit.
10	GRP 1 MATCH (Group 1 match) – Set to cause a group 1 match.
9	GRP 0 MATCH (Group 0 match) – Set to cause a group 0 match.
8	FORCE TMO RD (Forced timeout on read) – Set to cause a timeout on a read operation.
7:0	MBZ (Must be zero).

SBI Error Register The SBI error (SBIER) is a read-write register used to monitor and control error conditions. Figure 10-53 shows the register information format.

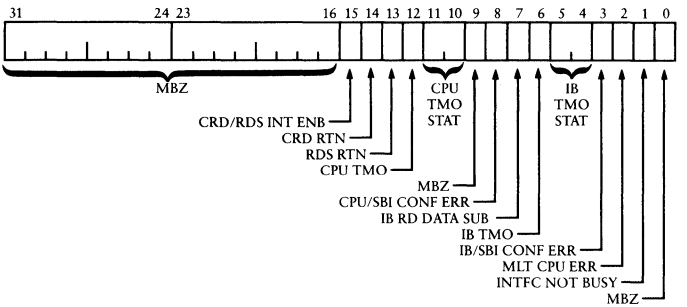


Figure 10-53 • SBI Error Register Format

Bit	Function
31:16	MBZ (Must be zero).
15	CRD/RDS INT ENB (CRD/RDS interrupt enable) – Set to enable a corrected read data or read data substitute command.
14	CRD RTN (CRD returned) – A write control bit set when the corrected read data is returned to the CPU.
13	RDS RTN (RDS returned) – A write control bit set when the read data substitute is returned to the CPU.
12	CPU TMO (CPU timeout) – A write control bit set to indicate that the CPU has timed out.
11:10	CPU TMO STAT (CPU timeout status) – Read-only bits.
9	MBZ (Must be zero).
8	CPU/SBI CONF ERR (CPU/SBI confirmation error) – A read-only bit set to indicate that the error between the CPU and SBI has been confirmed.
7	IB RD DATA SUB (IB read data substitute) – A write control bit set to cause a read data substitute.

Bit	Function
6	IB TMO (Information bus timeout) – A write control bit.
5:4	IB TMO STAT (IB timeout status) – A read-only field.
3	IB/SBI CONF ERR (IB/SBI confirmation error) – A read-only bit set to confirm that an error has occurred between the information bus and the CPU.
2	MLT CPU ERR (Multiple CPU error) – A read-only bit set to indicate that several CPU errors have occurred.
1	INTFC NOT BUSY (Interface not busy) – A read-only bit set to indicate that the SBI interface is not performing a transaction.
0	MBZ (Must be Zero).

SBI Timeout Address Register The SBI timeout address (SBITA) is a read-only register that stores the physical address sent on the SBI bus when a timeout occurs. Figure 10-54 shows the format of the information in the register.

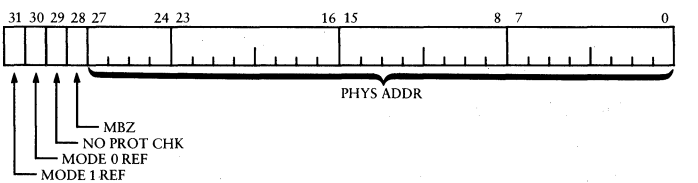


Figure 10-54 ▪ SBI Timeout Register Format

Bit	Function
31	MODE 1 REF (Mode 1 reference).
30	MODE 0 REF (Mode 0 reference).
29	NO PROT CHK (Protection not checked).
28	MBZ (Must be zero).
27:0	PHYS ADDR (Physical address) – A binary number that is the physical address sent to the SBI when the timeout occurred.

SBI Quadword Clear Register The SBI quadword clear (SBIQC) is a write-only register that contains the physical quadword address. Figure 10-55 shows the format of the register information.

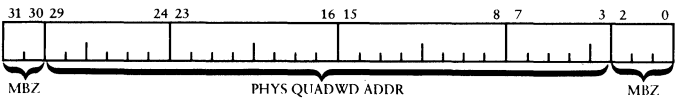


Figure 10-55 • SBI Quadword Clear Register Format

Bit	Function
31:30	MBZ (Must be zero).
29:3	PHYS QWDWD ADDR (Physical quadword address) – A binary number that is the physical quadword address.
2:0	MBZ (Must be zero).

• FP780 FLOATING-POINT ACCELERATOR

The FP780 floating-point accelerator option contains two internal registers that monitor and control the accelerator operation: the accelerator control/status (ACCS) register and the accelerator maintenance (ACCR) register.

Accelerator Control/Status Register The ACCS is a read-write register that indicates whether the accelerator option is installed. It controls when it is enabled, identifies its type, and reports errors and status functions. Figure 10-56 shows the format of the information in the ASSC register.

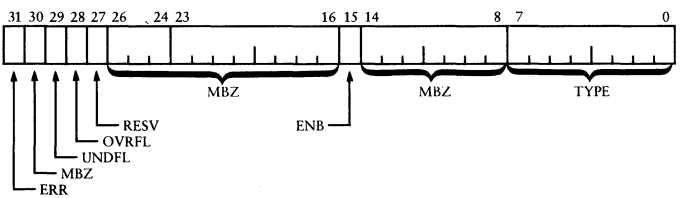


Figure 10-56 • FP780 Accelerator Control/Status Register Format

Bit	Function
31	ERR (Error) – A read-only bit set to indicate that one of the error bits (bits 30:27) is set.
30	MBZ (Must be zero).
29	UNDFL (Underflow) – A read-only bit set to indicate that the last instruction had an underflow caused by an exponent that was too small.
28	OVRFL (Overflow) – A read-only bit set to indicate that the last instruction resulted in an overflow caused by an exponent that was too large.
27	RESV (Reserved) – A read-only bit set to indicate that the last instruction had a reserved operand.
26:16	MBZ (Must be zero).
15	ENB (Enabled) – A read-write bit set during the bootstrap loading sequence to indicate that the accelerator is installed and enabled.
14:8	MBZ (Must be zero).
7:0	TYPE – A read-only field that specifies the FPA type as follows: 0 – No accelerator 1 – Accelerator installed 2 through 127 – Reserved for Digital Equipment Corporation 125 through 255 – Reserved for Digital's CSS group customers

Accelerator Maintenance Register The ACCR is a read-write register that controls the accelerator's microprogram counter (PC). The content of the register during bootstrap loading sequence is unpredictable. Figure 10-57 shows the format of accelerator maintenance register.

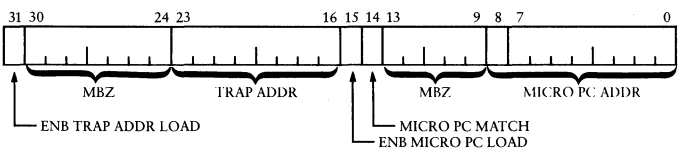


Figure 10-57 ▪ FP780 Accelerator Maintenance Register Format

Bit	Function
31	ENB TRAP ADDR LOAD (Enable trap address load) – A write-only bit set to cause bits 23:16 to be loaded into the FPA trap address register.
30:24	MBZ (Must be zero).
23:16	TRAP ADDR (Trap address) – A read-write field used by the main processor to force the accelerator to a specified micro location.
15	ENB MICRO PC LOAD (Enable micro PC load) – A write-only bit set to cause bits 8:0 to be loaded into the accelerator's micro PC match register.
14	MICRO PC MATCH (Micro PC match) – A read-only bit that is set whenever the accelerator's micro PC matches the micro PC match register.
13:9	MBZ (Must be zero).
8:0	MICRO PC ADDR (Micro PC address) – During a read operation, this field contains the next micro PC address to be executed. During a write operation, if EML (bit 15) is set, this value updates the micro PC match register.

VAX 8600 Processor-specific Registers

In addition to the architectural processor registers, the VAX 8600 processor contains the specific registers defined in this section and listed in table 10-5. Table 10-5 lists the register acronym, the hexadecimal processor address, and the register type.

Table 10-5 • VAX 8600 Processor-specific Registers

Register Name	Acronym	Address	Type*
Physical Address Memory Access	PAMACC	40	R/W
Physical Address Memory Location	PAMLOC	41	R/W
Cache Sweep	CSWP	42	R/W
Mbox Data ECC	MDECC	43	R/W
Mbox Error Enable	MENA	44	R/W
Mbox Data Control	MDCTL	45	R/W
Mbox MCC Control	MCCTL	46	R/W

* R/W is read or write, R is read-only, and W is write-only.

Register Name	Acronym	Address	Type*
Mbox Error Generator	MERG	47	R/W
Console Reboot	CRBT	48	W
Diagnostic Fault Insertion	DFI	49	W
Error Handling Status	EHSR	4A	R/W
Storage Transmit Control/Status	STXCS	4C	R/W
Storage Transmit Data Buffer	STXDB	4D	R/W
Ebox Scratchpad Address	ESPA	4E	W
Ebox Scratchpad Data	ESPD	4F	R

* R/W is read or write, R is read-only, and W is write-only.

• MEMORY REGISTERS

Several registers are provided to monitor information, to control the access to information in the cache and main memory, and to report errors when they occur. These registers are described in the following paragraphs.

Physical Address Memory Map Registers Two physical address memory map (PAMM) registers are included to allow information to be read from or written to the physical memory map locations: the physical address memory access (PAMACC) register and the physical address memory location (PAMLOC) register.

Physical Address Memory Access Register The physical address memory access (PAMACC) register is used to write to the memory locations and contains both the address of the location to be loaded and the data to be stored. Figure 10-58 shows the format of the data in the PAMACC register.

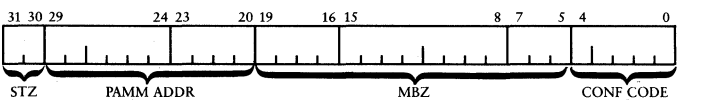


Figure 10-58 • Physical Address Memory Access Register Format

Bit	Function
31:30	STZ (Set to zero).
29:20	PAMM ADDR (Physical address memory map address) – Contains the PAMM address, which is the Ebox virtual address (EVA) bus (bits 29:20).
19:5	MBZ (Must be zero).
4:0	CONF CODE (Configuration code) – A hexadecimal code used to define the memory array card, adapter code, or nonexistent memory address as follows: **Code** **Configuration** 00 through 07 Internal array slot 0-7 respectively 18 through 1B I/O adapter 0 through 3 respectively 1F Nonexistent address

Physical Address Memory Map Location Register The physical address memory map location (PAMLOC) is a read-write register that specifies the location in the physical memory where the data will be read. When reading a location, PAMLOC is used to identify the location to be read and PAMACC is used to access the information. The data is returned in the same format as specified by the configuration code (bits 4:0) of the PAMMAC register. Figure 10-59 shows the information contained in the PAMLOC register.

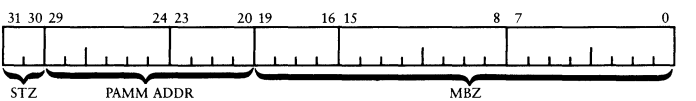


Figure 10-59 • Physical Address Memory Map Location Register Format

Bit	Function
31:30	STZ (Set to zero).
29:20	PAMM ADDR (Physical address memory map address) – Contains the PAMM address, which is the Ebox virtual address (EVA) bus (bits 29:20).
19:0	MBZ (Must be zero).

Cache Sweep Register The cache sweep (CSWP) is a read-write register used by the macrocode to monitor and control the state of the cache memory. During a powerup condition when the system is initialized, a cache sweep and invalidate operation are performed. During a powerdown condition, a validate operation is performed to write the cache information into memory. During the initialization procedure, the cache information is invalidated by clearing the data and valid bits. When half of the cache memory is disabled because of an error or when the state of the cache is accessed, the cache information is validated by copying the written cache locations into memory as specified by the cache-enable bits of the CSWP. The cache state can be changed by executing a MTPR instruction with the cache-control field value set to the required function. The current cache state can be obtained be executing a MFPR instruction and monitoring bits 1 and 0 of the CSWP register. The format of the register information is shown in figure 10-60.

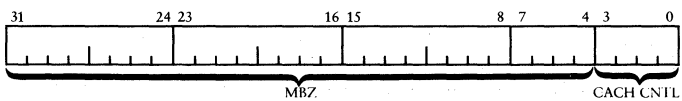


Figure 10-60 ▪ Cache Sweep Register Format

Bit	Function
31:4	MBZ (Must be zero).
3:0	CACH CNTL (Cache control) – The cache control field is defined as follows: Bit 3 – When set, the information in both halves of the cache memory will be cleared, including the written and valid bits, without updating memory. The cache-access mode is then enabled to select one or both of the halves of the cache memory as determined by the C1 ENB and C0 ENB bits. Bit 2 – When set, a copy of the information in one or both cache memory halves, as specified by the C1 ENB and C0 ENB bits, is written to memory. The cache-access mode is then enabled to select one or both of the halves of the cache memory, as determined by the C1 ENB and C0 ENB bits. Bit 1 – Set to enable the operation the C1 half of the cache memory. Bit 0 – Set to enable the operation the C0 half of the cache memory.

Mbox Data ECC Register The Mbox data ECC (MDECC) is a read-write register that contains the error flags related to the error correction code (ECC) and syndrome indicators that are used to correct single-bit errors in the cache or main memory. The register also contains other syndrome indicators and are used by the diagnostic programs and the macrocode to verify the system error-handling procedures. Figure 10-61 shows the function of the bits in the MDECC register.

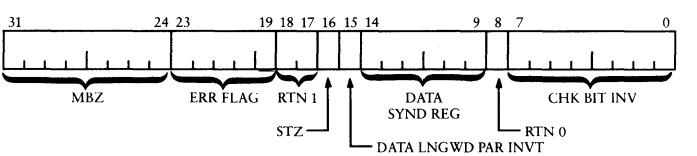


Figure 10-61 • Mbox Data ECC Register Format

Bit	Function
31:23	MBZ (Must be zero).
22:19	ERR FLG (Error flags) – Set to indicate a single-bit cache or memory error has occurred. The bits are described as follows: Bit 22 – A read-only bit set to indicate that incorrect data has been written into the cache or memory array from an external source. Bit 21 – A read-only bit set to indicate that the data read from the cache or main memory has a single-bit error. The DATA SYND REG (bits 14:9) contain the syndrome indication of this error. Bit 20 – A read-only bit set to indicate that the data word read from cache or main memory has a double-bit error or a detectable multiple-bit error. Bit 19 – A read-only bit set to indicate that the word fetched from memory was written to or read from the wrong location.
18:17	RTN 1 (Returned as one).
16	STZ (Set to zero).
15	DATA LNGWD PAR INVT (Data long parity inverted) – A read-only bit, set to indicate that the longword parity generated by the ECC logic is inverted.
14:9	DATA SYND REG (Data syndrome register) – Read-only bits; the syndrome generated by the ECC logic for the word in error.
8	RTN 0 (Returned as zero).
7:1	CHK BIT INVT (Check bits inverted) – Read-modify-write bits used when the ENB INV REG (bit 9) of the MDCTL register is set, to initiate ECC errors and A bus parity errors. These bits are set by the diagnostic program to perform the following functions: Bit 7 – The check parity bit 0 is inverted. Bits 6:1 – Check bits 6 through 1 respectively are inverted. Bit 0 – A write-only bit that is set to invert the longword parity, generated on the memory array bus, when the ECC logic is in the check mode.

Mbox Error Enable Register The Mbox error enable (MENA) is a read-write register used to enable the reporting of the Mbox error status bits. The associated traps and errors that are reported will still occur. This register is normally initialized by the console subsystem during the bootstrap sequence and is used for diagnostic and error-handling verification. Figure 10-62 shows the format of the information in the MENA register.

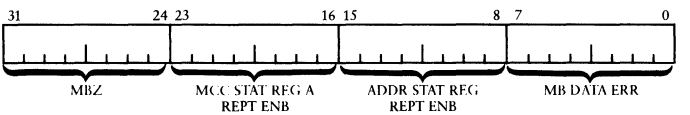


Figure 10-62 • Mbox Error Enable Register Format

Bit	Function
31:24	MBZ (Must be zero).
23:16	MCC STAT REG A REPT ENB (MCC status register A reporting enable)–A hexadecimal value (F8) used to enable the reporting of an error in the Mbox cache control status register, as follows: Bit 23–Set to indicate that a CPR parity error B has occurred. Bit 22–Set to indicate that a CPR parity error A has occurred. Bit 21–Set to enable the reporting of an A bus data parity error. Bit 20–Set to indicate that an A bus control parity error has occurred. Bit 19–Set to indicate that an A bus address parity error has occurred. Bits 18:16–Not used.
15:8	ADDR STAT REG REPT ENB (Address Status Register Reporting Enable)–A hexadecimal number (0F) used to enable the reporting of errors in the address status register in the Mbox, as follows: Bits 15:12–Not used. Bit 11–Set to indicate that a TB valid error has occurred. Bit 10–Set to indicate that a PTE B parity error has occurred. Bit 9–Set to indicate that a PTE A parity error has occurred. Bit 8–Set to indicate that a TB tag parity error has occurred.

Bit	Function
7:0	MB DATA ERR (Mbox data error) – A hexadecimal value (F9) used to enable the reporting of Mbox data errors, as follows: Bit 7 – Set to indicate that a write-byte-3 parity error has occurred. Bit 6 – Set to indicate that a write-byte-2 parity error has occurred. Bit 5 – Set to indicate that a write-byte-1 parity error has occurred. Bit 4 – Set to indicate that a write-byte-0 parity error has occurred. Bit 3 – Set to indicate that a cache data parity error has occurred. Bits 2:1 – Not used. Bit 0 – Set to indicate that cache-to-processor byte-write-data parity error has occurred.

MBox Data Control Register The Mbox data control (MDCTL) is a read-write register that is used by the diagnostic program to verify system error-handling procedures. Figure 10-63 shows the format of the information in the MDCTL register.

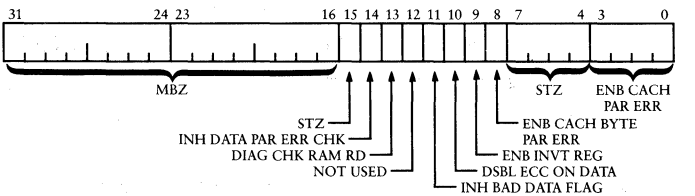


Figure 10-63 ▪ Mbox Data Control Register Format

Bit	Function
31:16	MBZ (Must be zero).
15	STZ (Set to zero).
14	INH DATA PAR ERR CHK (Inhibit data parity error check) – When set, the MD bus write parity errors and the cache data parity errors will not be detected or reported by the Mbox.
13	DIAG CHK RAM RD (Diagnostic check RAM read) – When set, the bits from the selected cache RAMs will be read.

Bit	Function
12	Not used.
11	INH BAD DATA FLG (Inhibit bad data flag) – When set the the bad data code check bits will not be generated as described in the MDECC register.
10	DSBL ECC ON DATA (Disable ECC on data) – When set, no error correction on the data will occur. Error information will be logged and reported, unless inhibited by the MERG register.
9	ENB INVT REG (Enable invert register) – When set, the data stored in the data-check-bit invert register complements the check bits generated on the write.
8	ENB CACH BTYE PAR ERR (Enable cache byte parity error) – When set, the bad-byte parity will be written into cache from a CPU request as determined by the ENB CACH PAR ERR (bits 3:0).
7:4	STZ (Set to zero).
3:0	ENB CACH PAR ERR (Enable cache parity error) – When set in conjunction with the EN CACH BYTE PAR ERR (bit 8) even parity will be generated on bytes 3:0 and written into the cache during CPU write operations. These bits are also used to generate single longword failures during A bus write operations.

Mbox Memory Cache Control Register The Mbox memory cache control (MCCTL) is a read-write register used by the diagnostic programs and the macrocode for error-handling verification. Figure 10-64 shows the format of the information in the register.

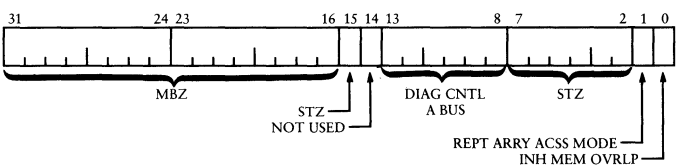


Figure 10-64 • Mbox Memory Cache Control Register Format

Bit	Function
31:16	MBZ (Must be zero).
15	STZ (Set to zero).
14	Not used.
13:8	DIAG CNTL ABUS (Diagnostic control A bus)–Used for maintenance when the Mbox is using the A bus in diagnostic mode. The bits stored in these locations are substituted for the normal data length/status bits and command/mask bits when writing to the I/O adapter register file. The bit assignments are as follows: Bits 13:12–Data length/status bits 1 and 0 are set. Bits 11:8–Command/mask bit 3 through 0 are set.
7:2	STZ (Set to zero).
1	REPT ARRY ACSS MODE (Repeatable array access mode–When set, the memory access time from the array is repeatable and slower than the normal access time plus the refresh time.
0	INH MEM OVRLP (Inhibit memory overlap)–When set, the data from a memory array cannot be read before the previous data read from memory has been extracted from the array register file.

Mbox Error Generator Register The Mbox error generator (MERG) is a read-write register used by the diagnostic programs and the verification macrocode to cause errors and verify the correction process. The VMS software can affect the error reporting process by setting appropriate bits in this register when error rates are exceeded. Figure 10-65 shows the bit assignments in this register.

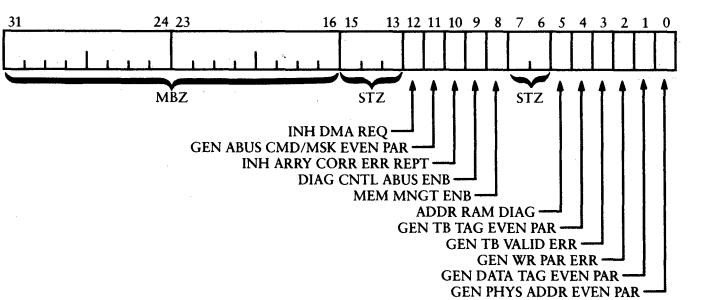


Figure 10-65 • Mbox Error Generator Register Format

Bit	Function
31:16	MBZ (Must be zero).
15:13	STZ (Set to zero).
12	INH DMA REQ (Inhibit DMA request) – When set, the DMA requests from the SBI adapter will not be serviced.
11	GEN ABUS CMD/MSK EVEN PAR (Generate A bus command/mask even parity) – Set to cause the A bus command and mask to be sent with even parity.
10	INH ARRY CORR ERR REPT (Inhibit array correctable error reporting) – Set to prevent the reporting of correctable errors when reading the main memory. The errors are corrected by the logic.
9	DIAG CNTL ABUS ENB (Diagnostic control enable) – When set, the diagnostic-control A bus information (bits 13:8 of the MCCTL register) is used on the A bus.
8	MEM MNGT ENB (Memory management enable) – Set to enable the memory management function that allows the virtual addresses to be translated by the translation buffer. When cleared, all memory references will be recognized as physical and the parity error checking of the translation buffer will be disabled.
7:6	STZ (Set to zero).
5	ADDR RAM DIAG (Address RAM diagnostics) – Set to cause special events to occur in the Mbox depending on the type of cycle, as follows: Write TB – The translation buffer RAMs that contain the physical address (bits 12:9) are selected by the IVA bus to be written instead of by the EVA bus. Read TB – The contents of the translation buffer RAMs containing physical address bits (12:9) are selected by the IVA bus to be read instead of the EVA bus. Diagnostic read cache tag address – The written bit, written parity valid bit, and cache tag parity will be read instead of the cache tag address bits.
4	GEN TB TAG EVEN PAR (Generate TB tag even parity) – When set, data written to the TB will include a TB tag with even parity. A parity error will be indicated when a read data command specifies this location.

Bit	Function
3	GEN TB VALID ERR (Generate TB valid error) – When set, both stored valid bits will be written to the same value.
2	GEN WR PAR ERR (Generate written parity error) – When set, the written parity bit in the cache tag is complemented before being written into cache.
1	GEN DATA TAG EVEN PAR (Generate data tag even parity) – When set, the cache data address tag is stored with even parity. The normal parity is the odd parity generated on the stored tag address and the four valid bits.
0	GEN PHYS ADDR EVEN PAR (Generate physical address even parity) – When set, the cache data address tag included in the data-path-check bit generation is complemented and written either into cache or into the storage array.

▪ DIAGNOSTIC REGISTERS

Two registers are provided for use by the diagnostic programs and error handling macrocode to verify processor errors. These registers are described in the following paragraphs.

Diagnostic Fault Insertion Register The diagnostic fault insertion (DFI) is a write-only register used by the diagnostic program and the macrocode to verify the handling of errors. Figure 10-66 shows the format of the DFI register.

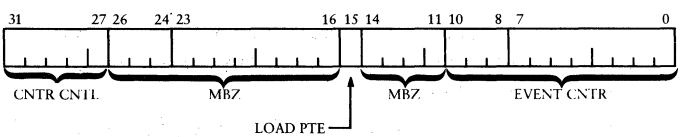


Figure 10-66 ▪ Diagnostic Fault Insertion Register Format

Bit	Function
31:27	CNTR CNTL (Counter control) – Used to select the operating conditions of the DFI register as follows: Bit 31 – Set to specify that the fault insertion is enabled and controlled by the hardware associated with this register. Bit 30:28 – These bits form the count mode 0-2 for the event counter increment enable as described: **Bit 30 / Bit 29 / Bit 28 / Event** 0 0 0 No count 0 0 1 Count 0 1 0 -Estall 0 1 1 -Istall 1 0 0 Ebox Umark + Ibox Umark 1 0 1 Ebox Umark * -Estall + Ibox Umark * -Istall 1 1 0 IRD * - stall Bit 27 – When set, the microcode mark is overridden as one of the conditions included to insert the fault.
26:16	MBZ (Must be zero).
15	LOAD PTE (Load page-table entry) – When set, the page-table entry in general purpose register R1 is loaded into the TB location corresponding to the virtual address in register R0 and all other bits in this register are ignored.
14:11	MBZ (Must be zero).
10:0	EVENT CNTR (Event counter) – A binary counter that specifies the count value of from one to 1,047 cycles. The count values are selected by bits 30:28 and generated by Ebox and Ibox microcode mark bits with and without stalls, IRD cycles not stalled, or an external event until the event counter overflows.

Error Handling Status Register The error handling status register (EHSR) is a read-write register used by the error-handling microcode (EHM) and macrocode for the synchronization and control of the machine-check exception processing. This register is part of the reason code field of the machine-check exception stack frame. Figure 10-67 shows the format of the information in the register.

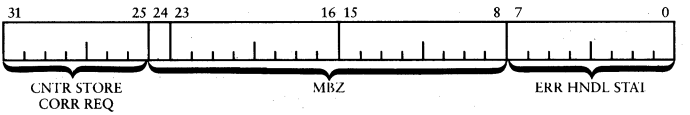


Figure 10-67 ▪ Error Handling Status Register Format

Bit	Function
31:25	CNTL STORE CORR REQ (Control store correction required) – Read-only bits set to indicate the request that was generated by the EHM to the console as follows: Bit 31 – Set to indicate that a correction is required to the Fbox multiplier control store microcode. Bit 30 – Set to indicate that a correction is required to the Fbox adder control store microcode. Bit 29 – Set to indicate that a correction is required to the Fbox DRAM microcode. Bit 28 – Set to indicate that a correction is required to the Ibox DRAM microcode. Bit 27 – Set to indicate that a correction is required to the the Ibox control store microcode. Bit 26 – Set to indicate that the correct copy of the Mbox error address is contained in the Mbox scratchpad memory. Bit 25 – Set to indicate that the error-handling microcode has determined that an unrecoverable error has occurred and the operating system will be automatically reloaded.
24:8	MBZ (Must be zero).
7:0	ERR HNDL STAT (Error handling status) – Indicates the status of the error-handling routine as follows: Bit 7 – Set to indicate that the EHM has been notified by the interrupt and exception microcode to service an Mbox error. Bit 6 – Set to indicate that the EHM has been entered. Bit 5 – Set to indicate that the program count of the machine check exception handler has been transferred to the VMS operating system. Bit 4 – Set to indicate that EHM has been informed of an Fbox hardware error. The EHM will perform a control store or GPR correction if required.

Bit	Function
	Bit 3 – Set to indicate that the EHM routine attempted to correct the error with the Fbox GPR RAMs.
	Bit 2 – Set to indicate that the EHM attempted to correct the error with the A side of the Ebox scratchpad RAMs.
	Bit 1 – Set to indicate that the EHM attempted to correct the error with the B side of the Ebox scratchpad RAMs.
	Bit 0 – Set to indicate that the EHM attempted to correct the error with the Ibox GPR RAMs.

• CONSOLE SUBSYSTEM INTERFACE

The console subsystem interface contains registers used to transfer information between the VAX 8600 CPU and the external devices that connect to the console subsystem. Two console transmit registers transfer information from the CPU to the devices and two console receive registers transfer information from the devices to the CPU. The console reboot register enables the VMS operating system to reboot the console program after a console failure. The external devices that communicate with the CPU are the console terminal, remote diagnostic port, environmental monitoring module (EMM), and RL02 console load disk drive.

Console Reboot Register The console reboot (CRBT) is a write-only register controlled by software to reboot the console processor only. The register contains the reboot code and address supplied by the microcode. Figure 10-68 shows the format of the information in the register.

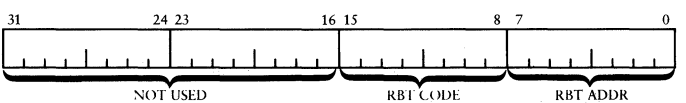


Figure 10-68 • Console Reboot Register Format

Bit	Function
31:16	Not used.
15:8	RBT CODE (Reboot code) – The reboot code is 80 (hexadecimal).
7:0	RBT ADDR (Reboot address) – The reboot address is 33 (hexadecimal) and causes a bootstrap loading of the console processor.

Console Receive Control/Status Register The console receive control/status (RXCS) is a read-write register used with the console receive data buffer register (RXDB) to transfer information from the console subsystem input to the CPU. Figure 10-69 shows the format of the information in the RXCS register.

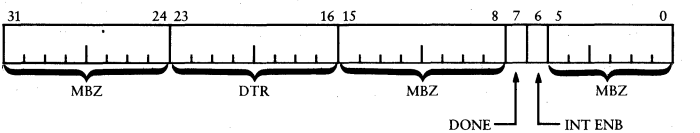


Figure 10-69 ▪ Console Receive Control/Status Register Format

Bit	Function
31:24	MBZ (Must be zero).
23:16	DTR (Data terminal ready) – Read-write bits set by CPU to indicate that the CPU is ready to accept characters from the modem. Cleared to indicate that the CPU is ready to disconnect from the modem. The DTR assignments are as follows: Bits 23:20 – Reserved Bit 19 – Logical console subsystem data Bit 18 – EMM data line Bit 17 – Remote services port Bit 16 – Console terminal
15:8	MBZ (Must be zero).

Bit	Function
7	DONE – A read-only by CPU bit set to indicate that data is available from the console subsystem and that the data and the source identification (ID) is available in the RXDB register.
6	INT ENB (Interrupt enable) – A read-write bit set by the CPU to allow an interrupt request to be generated if the DONE (bit 7) is also set.
5:0	MBZ (Must be zero).

Console Receive Data Buffer Register The console receive data buffer (RXDB) is a read-only register that receives data from the console subsystem for transfer to the CPU. It is initialized by the console subsystem according to the CPU control panel switch settings and the devices connected to the console subsystem. Figure 10-70 shows the format of the information in the register.

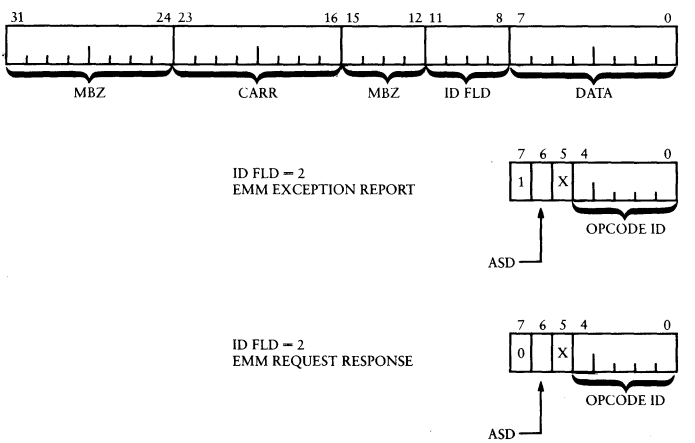


Figure 10-70 • Console Receive Data Buffer Register Format

Bit	Function
31:24	MBZ (Must Be zero).
23:16	CARR (Carrier)–A CPU read-only field that is initialized by the console subsystem to a value based on the state of the corresponding DTR bit in the RXCS register.
15:12	MBZ (Must be zero).
11:8	ID FLD (Identification field)–A CPU read-only field that contains a hexadecimal value to identify the source of data in this register, as follows: Value 0–The data in bits 7:0 of this register is from the console terminal. Value 1–The data in bits 7:0 of this register is from the remote diagnostic line. Value 2–The data in bits 7:0 of this register is an opcode from the EMM. Value 3–The data in bits 7:0 of this register is related to the logical console line. Value 4 through OE–Reserved for Digital Equipment Corporation. Value OF–CARR (bit 23:16) has changed.
7:0	DATA–The data/status information as specified by the ID (bits 11:8).

Console terminal (ID FLD = 0)–Console terminal data is specified by an identification field value of 0. The data received on the console terminal is contained in the data byte (bits 7:0) of the RXDB.

Remote services port (ID FLD = 1)–Remote service port data is specified by an identification field value of 1. The data received on the remote services port is contained in the data byte (bits 7:0) of the RXDB.

Environmental Monitoring Module (ID FLD = 2)–The environmental monitoring module data is specified by an identification field value of 2. The RXDB data byte (bits 7:0) contains an opcode from the environmental monitoring module (EMM) that defines the information in the data byte that follows the opcode. The two types of information that can be transferred are unsolicited exception reports from the EMM and responses from

the EMM as a result of requests. The opcode formats for the EMM data, shown in figure 10-70, are defined as follows:

- *EMM exception reports* – These reports are contained in a single opcode byte followed by a single data byte. The format of information in the opcode byte is described as follows:

  Bit 7 – Set to specify an EMM exception report.

  Bit 6 – Set to indicate that total system power shutdown is pending.

  Bit 5 – Not used.

  Bits 4:0 (Opcode ID) – A hexadecimal number that identifies the type of information in the data byte.

  – Power supply status – The power supply voltage regulators in the CPU are monitored by the EMM. The status of the regulators' output voltage is indicated by the opcode ID (bits 4:0) values that follow. The data byte (bits 7:0) following the opcode byte indicates the present condition of the regulator voltage.

Data byte value	Status	
0	Voltage is normal	
1	Voltage is not within specifications	
Opcode ID	**Regulator**	**Voltage**
0	A	+5 V
1	B	+5 V
2	C	+5 V
3	D	−2 V
4	E	−2 V
5	F	−5.2 V
6	H	−5.2 V
7	L	+12 V
8	L	−12 V
9	K	+15 V
A	K	−15 V

  – Temperature status – The temperatures in the CPU cabinet are monitores by sensors. The status of the temperature is indicated by the opcode ID (bits 4:0) values that follow. The data byte (bits 7:0) following the opcode byte indicates the present condition of the temperature.

Data byte value	Status
0	Temperature is within normal range
1	Temperature is in yellow zone
2	Temperature is in red zone and power will be removed from the system if condition is not corrected.
3	Temperature is below normal range

Opcode ID	Temperature
B	T1
C	T2
D	T3
E	T4
F	T2-T1 delta change
10	T3-T1 delta change
11	T4-T1 delta change

– Air flow status – Sensors in the CPU cabinet monitor the flow of cooling air. The status of the air flow is indicated by the opcode ID (bits 4:0) values that follow. The data byte (bits 7:0) following the opcode byte indicates the present condition of the air flow.

Data byte value	Status
0	Air flow is normal
1	Air flow is not within specifications and power will be removed from the system if condition is not corrected.

Opcode ID	Sensor
12	1
13	2

– Battery backup status – The battery backup unit (BBU) provides power to the memory and other critical functions in the event of a power interruption. The status of the battery backup indicated by the opcode ID (bits 4:0) value that follows. The data byte (bits 7:0) following the opcode byte indicates the present condition of the BBU.

Data byte value	Status
0	BBU is available
1	BBU is not available
Opcode ID	**Status**
14	BBU available status

– EMM status – The status of the EMM is monitored and failures in the module operation are reported. An EMM failure is indicated by the opcode ID (bits 4:0) value that follows. The data byte (bits 7:0) following the opcode byte indicates the type of EMM failure.

Data byte value	Status
0	Failed to restart
1	Detected an EMM RAM parity error
2	Detected an illegal instruction
3	Detected an unknown trap to zero
4	Detected an unexpected trap interrupt
5	Detected an unexpected 6.5 interrupt
6	Excessive collision on the EMM bus
7	No transport acknowledge from the EMM
8	No response from EMM
9	Negative response from the EMM
A	No EMM buffers available
B	Console to EMM message transmit timeout
Opcode ID	**Status**
15	EMM failure

– Transmit read time-out – The ready (bit 7) in the TXCS register is monitored for a timeout condition. If the bit is not set by the console within two seconds, the transmit operation that was in process is terminated. The timeout condition is indicated by the opcode byte (bits 4:0) value that follows. The data byte (bits 4:0) following the opcode byte indicates the timeout condition.

Data byte value	Status
0	Local terminal operation is aborted
1	Remote services port operation is aborted
2	EMM operation is aborted
3	Logical console operation is aborted
Opcode ID	**Status**
16	Transmit ready timeout

- *EMM Request Response*–EMM responses that are solicited can vary in length. The first data byte contains the opcode that defines the information. The second data byte defines the number of data bytes that will follow. The two types of responses that are available are the EMM status response and the EMM environmental response. The format of the EMM request opcode byte follows:

  Bit 7–Cleared to specify an EMM request response.

  Bit 6–Set to indicate total system power shutdown is pending.

  Bit 5–Not used.

  Bits 4:0 (Opcode ID)–A hexadecimal number that indicates the type of information contained in the EMM status response or an EMM evironmental response.

  –*EMM status response*–An EMM status response returns the status of the EMM unit and is specified by an opcode ID (bits 4:0) value of 0. The first data byte following the opcode byte has a value 8 indicating the number of data bytes that will follow. The status information contained in the individual bits of the data bytes (bits 7:0) are:

Byte 1–Power control register

Bit	Regulator	Status
0	B	0 = off (+5 V BBU)
1	C	0 = off (+5 V SBIA)
2	D	0 = off (−2 V ECL)
3	E	(follows state of bit 2)
4	F	0 = off (+5.2 V ECL)
5	H	(follows state of bit 4)
6	J	Not used
7	BBU	Disable status (0 = enabled)

Byte 2 – Margin Enable register

Bit	Regulator	Status
0	A	0 = nominal
1	B	0 = nominal
2	C	0 = nominal
3	D	0 = nominal
4	E	(follows bit 3)
5	F	0 = nominal
6	H	(follows bit 5)
7	J	0 = nominal

Byte 3 – Margin select register

Bit	Regulator	Status
0	A	0 = low, 1 = high
1	B	0 = low, 1 = high
2	C	0 = low, 1 = high
3	D	0 = low, 1 = high
4	E	(follows bit 3)
5	F	0 = low, 1 = high
6	H	(follows bit 5)
7	J	0 = low, 1 = high

Byte 4 – Least significant byte of the MODOK register

Bit	Regulator	Status
0	A	(OK) 1 = OK
1	B	(OK) 1 = OK
2	C	(OK) 1 = OK
3	D	(OK) 1 = OK
4	E	(OK) 1 = OK
5	F	(OK) 1 = OK
6	H	(OK) 1 = OK
7	J	(OK) 1 = OK

Byte 5 – Most significant byte of the MODOK register

Bit	Regulator	Status
0	K	1 = OK
1	L	1 = OK
2	K	(ac low) 1 = OK
3	L	(ac low) 1 = OK
4		EMM unit number = 0
5		
6		
7	J	Key override circuit

Byte 6 – Miscellaneous hardware status register

Bit	Function	Status
0	Air flow 1 sensor	1 = fault
1	BBU	1 = failure
2	−2 V crobar signal	1 = crobar
3	Air flow 2 sensor	1 = fault
4	Latched ac low signal	1 = ac low
5	Latched dc low signal	1 = dc low
6	Parity checker	1 = OK
7	Parity error	0 = OK

Byte 7 – Miscellaneous software status register

Bit	Function	Status
0	Extended output signal	1 = asserted
1	Default mode enable	1 = asserted
2	Auto shutdown	1 = active
3	5.5 interrupt disable	1 = disabled
4:7	Unused	

Byte 8 – Integer value of the EMM PROM version

– *EMM environment response* – An EMM environment response returns all measured environmental information and is specified by an opcode ID (bits 4:0) value of 1. The first data byte following the opcode byte has a value of 32 indicating the number of data bytes that will follow. The voltage values are digitized and must be converted to the actual value before being displayed or interpreted. The conversions apply to the regulator and thermistor voltages bytes that follow:

(1) **Volts** = 0.0276 μ magnitude
(2) **Milliamperes** = 70 + (2.8 μ magnitude)
(3) **Volts** = 0.0762 μ magnitude
(4) **Volts** = 0.0691 μ magnitude
(5) **Degrees Celsius** = 16 + (0.3333 μ magnitude)

Regulator voltage – Data byte 1 through 24 are regulator voltage byte-pairs. The first data byte of a pair indicates the voltage magnitude. In the second data byte, bits 0:6 are reserved and bit 7 indicates the voltage polarity; 1 = positive, 0 = negative. The regulators assignments are:

Byte	Regulator	Conversion
1	A (+5 V TTL)	(1)
2	A (polarity)	
3	B (+5 V BBU)	(1)
4	B (polarity)	
5	C (+5 V SBIA)	(1)
6	C (polarity)	
7	D (−2 V ECL)	(1)
8	D (polarity)	
9	E (−2 V ECL)	(1)
10	E (polarity)	
11	F (−5.2 V ECL)	(1)
12	F (polarity)	
13	H (−5.2 V ECL)	(1)
14	H (polarity)	
15	Ground current voltage	(2)
16	Ground current polarity	
17	L (+12 V TTL)	(3)
18	L (polarity)	
19	L (−12 V TTL)	(4)
20	L (polarity)	
21	K (+15 V TTL)	(4)
22	K (polarity)	
23	K (+15 V TTL)	(4)
24	K (polarity)	

Thermistor voltage—Data byte 25 through 32 are thermistor voltage byte-pairs. The first data byte of a pair indicates the voltage magnitude. In the second data byte, bits 0:6 are reserved and bit 7 is 0 indicating a positive voltage polarity. The thermistor assignments are:

Byte	Thermistor	Conversion
25	T1 (input)	(5)
26	T1 (polarity)	
27	T2 (input)	(5)
28	T2 (polarity)	
29	T3 (output)	(5)
30	T3 (polarity)	
31	T4 (output)	(5)
32	T4 (polarity)	

Logical Console Data (ID FLD = 2)–Logical console data is specified by an identification field value of 3. The data byte (bits 7:0) of the RXDB can indicate any of the following:

- Data byte = 10–One byte of data will follow, indicating the status of the warm-start flag: 0 = warm flag cleared, 1 = warm flag set.
- Data byte = 11–One byte of data will follow, indicating the status of the cold-start flag: 0 = cold flag cleared, 1 = cold flag set.
- Data byte = 12–Two bytes of data will follow, indicating the microcode version number: byte 1 is the least significant 8-bits and byte 2 is the most significant 8-bits.
- Data byte = 20–Indicates that the console has successfully completed the the execution of the console command string.
- Data byte = (30)–One byte of data will follow indicating the condition of the snapshot files as follows:
  Bit 0 (0)–SNAP1.DAT file invalid
  Bit 0 (1)–SNAP1.DAT file valid
  Bit 1 (0)–SNAP2.DAT file invalid
  Bit 1 (1)–SNAP2.DAT file 2 valid
  Bits 7:2–Not used
- Data byte = 40–Indicates that the console has been sucessfully rebooted by VMS.
- Data byte = 82–Indicates that the console detected an error while executing the console command string, that the console command string exceeded the byte buffer space, or that console is disabled because of the control panel switch positions.

Console Transmit Control/Status Register The transmit control/status (TXCS) is a read-write register that operates with the transmit data buffer (TXDB) register to transfer information from the CPU to the console interface devices. The register is cleared by the processor during initialization. Figure 10-71 shows the format of the TXCS register information.

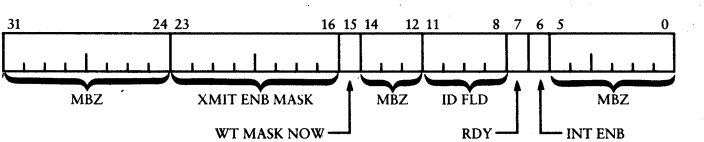


Figure 10-71 ▪ Console Transmit Control/Status Register Format

Bit	Function
31:24	MBZ (Must be zero).
23:16	XMIT ENB MASK (Transmit enable mask) – A read-write field where each bit is set to enable the transfer of information between the processor and a device, as follows: Bit 23:20 – Reserved for additional console-resident interfaces Bit 19 – Logical console subsystem data Bit 18 – EMM module data Bit 17 – Remote services port bit 16 – Console terminal
15	WT MASK NOW (Write mask now) – A CPU and console read-write bit, set by the CPU to allow the microcode to write to the transmit enable mask field (bit 23:16). The console reads this bit to determine the operation that caused the console interrupt.
14:12	MBZ (Must be zero).
11:8	ID FLD (Identification field) – A hexadecimal value that is read by the CPU to identify the line responsible for the interrupt and is ready to transmit data, as follows: Value 0 – Console terminal Value 1 – Remote services port Value 2 – EMM data Value 3 – Logical console data Value F – No lines currently enabled
7	RDY (Ready) – A CPU read-only bit set to indicate that the line identified in the identification field (bits 11:8) is ready to transmit data when the transmit enable mask field (bits 23:16) is not zero.
6	INT ENB (Interrupt enabled) – A CPU read-write bit set to enable an interrupt request from the device selected when the ready (bit 7) is also set.
5:0	MBZ (Must be zero).

Console Transmit Data Buffer Register The console transmit data buffer (TXDB) is a write-only register that receives the data to be transferred from the CPU to the selected device. Figure 10-72 shows the format of the register information.

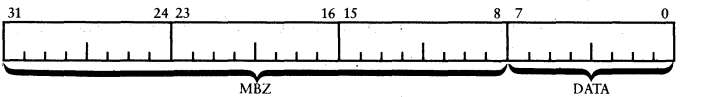


Figure 10-72 ▪ Console Transmit Data Buffer Register Format

Bit	Function
31:8	MBZ (Must be zero).
7:0	DATA – A CPU write-only byte of data for transmission to the selected console line as specified by the ID FLD (bits 11:8) of the TXCS register. The data and ID field values are as follows: ID value 0 – Data bytes for the console terminal line ID value 1 – Data byte for remote services port ID value 2 – Data byte for the EMM line to any one of the following hexadecimal values: **Byte 1 value** — **Function** 0 — Request for EMM status information 1 — Request system environment information 3 — Set regulator margins normal 4 — Set regulator margins low 5 — Set regulator margins high 11 — Cancel current and queued EMM requests ID value 3 – Data bytes for the logical console line; may contain any one of the following hexadecimal values: **Byte 1 value** — **Function** 2 — Initiate system reboot sequence 3 — Clear the console copy of the warm-start flag. 4 — Clear the console copy of the cold-start flag. 10 — Request the value of the console's copy of the warm-start flag. 11 — Request the value of the console's copy of the cold-start flag. 12 — Request the 16-bit value of the system microcode version being executed.

Bit	Function	
	Byte 1 value	**Function**
	20	Begin accepting a console commmand string. Used for error-handling verification.
	30	Request the console return the status of two snapshot files.
	31	Request the console to invalidate SNAP1.DAT file after processing SNAP2.DAT file.
	32	Request the console to invalidate SNAP2.DAT file after processing.
	70	Cancel all current and queued logical console requests.

- CONSOLE BLOCK STORAGE

The console subsystem contains two registers used to implement the data transfers between the CPU and the RL02 console disk drive. The CPU initiates the read and write operation and monitors the console status during the console block storage data transfers.

Storage Transmit Control/Status Register The storage transmit control/status (STXCS) is a read-write register used to monitor and control the transfer of information between the CPU and RL02 disk drive. It contains the logical block address of the disk where the data will be read or written. It also defines the read or write function and returns status to the processor during the transaction. Figure 10-73 shows the format of the information in the STXCS register.

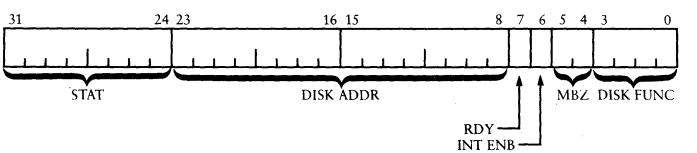


Figure 10-73 • Storage Transmit Control/Status Register Format

Bit	Function
31:24	STAT (Status) – A CPU read-only field containing a hexadecimal value that specifies the status of the current transfer as defined: Value 1 – Indicates that the console or the CPU has completed the data transfer operation. Value 2 – Set by the console or CPU to indicate that the entire block of data has not been transferred. Value 3 – Set by the console to indicate that the requested transfer has been aborted. Value 4 – Set by the CPU during a transfer operation to request status information from the console through the STXDB register. Value 80 – Set by the CPU during a transaction to indicate that a protocol error has occurred. Value 81 – Set by the CPU to indicate that a hardware error has been detected during the transaction.
23:8	DISK ADDR (Disk address) – A CPU write-only field that contains the hexadecimal address of the block number of the RL02 disk for the current transfer.
7	RDY (Ready) – A CPU read-only bit set to indicate that the register is ready for a transaction.
6	INT ENB (Interrupt enable) – A CPU write-only bit set to enable a data transfer to occur when the RDY (bit 7) is set.
5:4	MBZ (Must be zero).
3:0	DISK FUNC (Disk function) – A CPU write-only field that specifies the disk function to be performed, as follows: 0 = No operation 1 = Not assigned 2 = Reset and return device status 3 = Abort current transaction 4 = Read device status 5 = Write block data 6 = Read block data

Storage Transmit Data Buffer Register The storage transmit data buffer (STXDB) is a read-write register that contains the data to be read from or written to the console storage disk drive. The format of the information in the register is shown in Figure 10-74.

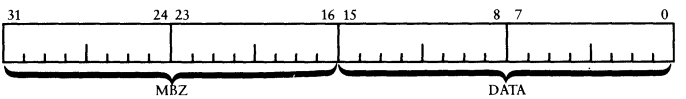


Figure 10-74 • Storage Transmit Data Buffer Register Format

Bit	Function
31:16	MBZ (Must be zero).
15:0	DATA–Contains the data to be transferred.

Ebox Scratchpad Registers The Ebox contains two dual-ported scratchpad memories that are accessed by the Ebox scratchpad registers. The Ebox scratchpad address (ESPA) register and the Ebox scratchpad data (ESPD) register are used by VMS machine check code to access Mbox status information when single-bit errors are processed in the memory array. The status information is saved by the error-handling microcode in the Ebox scratchpad memory locations 25 through 2B (hexadecimal).

Ebox Scratchpad Address Register The ESPA is a write-only register that contains the address of an Ebox scratchpad memory location to be accessed. Figure 10-75 shows the format of the register information.

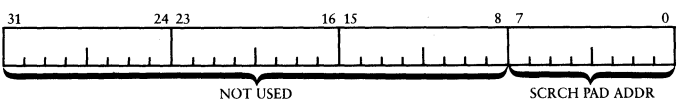


Figure 10-75 • Ebox Scratchpad Address Register Format

Bit	Function
31:8	Not used.
7:0	SCRCH PAD ADDR (Ebox scratchpad address)—Contains the address from 25 to 2B (hexadecimal) of the Ebox scratchpad location to be accessed.

Ebox Scratchpad Data Register The ESPD is a read-only register that contains the error status information in the Ebox scratchpad memory. Figure 10-76 shows the format of the data in the register.

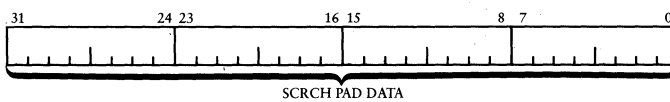


Figure 10-76 ▪ Ebox Scratchpad Data Register Format

Bit	Function
31:0	SCRCH PAD DATA (Ebox scratchpad data)—Contains the error status information saved in the Ebox scratchpad memory.

▪ FP86 FLOATING-POINT ACCELERATOR

The FP86 floating-point accelerator option contains an internal register to control and monitor the accelerator operation as described.

Accelerator Control/Status Register The accelerator control/status (ACCS) is a read-write register that controls the operation and provides status information on the FP86 floating-point accelerator option (Fbox). Figure 10-77 shows the format of the information in the register.

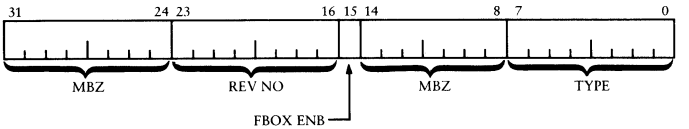


Figure 10-77 • FP86 Accelerator Control/Status Register Format

Bit	Function
31:24	Not used.
23:16	REV NO (Revision number) – The revision number of the FP86 or the floating terminator module (FJM) and floating jumper module (FJM) when the accelerator is not installed.
15	FBOX ENB (Fbox Enable) – Enables the Fbox to allow the execution of floating-point instructions.
14:8	Not used.
7:0	TYPE – A value that specifies the type of accelerator as follows:
	Value 1 – FP86 installed Value 0 – FP86 not installed

Chapter 11 · VAX Input/Output Subsystems

The VAX processors connect to devices, networks, and other VAX processors through the input/output (I/O) subsystems. These subsystems electrically connect the devices and processors to memory through the internal processor bus of each VAX processor. Several I/O subsystems are available. The number and type of subsystems that can be used depends on the VAX processor type, communication speed, and the peripheral devices or systems to be connected. The subsystems permit fast and reliable transfer of information. This chapter describes the general functions and operation of each subsystem.

To simplify the reference to VAX processors in this chapter, the processors are grouped into categories by physical size and computing power, as follows:

Processor Type	System
Small	VAX-11/725 and VAX-11/730.
Medium	VAX-11/750.
Large	VAX-11/780, VAX-11/782, VAX-11/785, and VAX 8600.

· I/O Subsystem Allocations

Four I/O subsystems are available for implementation with VAX processors. They are the UNIBUS subsystem, the MASSBUS subsystem, the Computer Interconnect (CI) subsystem, and the DR32 Device Interconnect (DDI) subsystem.

The UNIBUS subsystem consists of a UNIBUS adapter (UBA) and an asynchronous, bidirectional bus. It can communicate with many different types of intelligent and nonintelligent interfaces and devices. The subsystem supports a large number of standard Digital devices developed for both PDP-11 processors and VAX processors. It permits low- and medium-speed data transfers with the connected devices. Through the use of direct memory access (DMA) capability of the UNIBUS subsystem, blocks of data can be transferred to memory quickly and efficiently.

The MASSBUS subsystem consists of the MASSBUS adapter (MBA) and a high-speed, synchronous communication link between mass storage devices and the host processor. Blocks of data are transferred to the processor memory through the MASSBUS subsystem without the intervention of the processor at rates of up to 2 Mbytes per second. The MASSBUS device registers are addressed the same way as processor memory locations.

The Computer Interconnect (CI) subsystem consists of a CI adapter and a high-speed, dual-path, communication link between VAX processors and between VAX processors and intelligent mass storage subsystems. The CI subsystem operates with the VAXcluster architecture to allow sharing of processing facilities and mass storage resources. Data is transferred in packets through high-speed, dual-path coaxial cables. The CI subsystem allows VAX processors to be added to the VAXcluster architecture when required and data can be shared by any of the processors.

The DR32 Device Interconnect (DDI) subsystem is a high-speed, 32-bit, parallel, data communication path between Digital- or customer-developed devices. The DDI consists of an interface adapter and DDI bus and permits transfer of high-resolution information at rates of up to 6.67 Mbytes per second. Both command and data chaining are used to enable multiple I/O operations to occur without software intervention.

Table 11-1 lists the type and quantity of I/O subsystems that can be implemented on each VAX processor. The UNIBUS I/O subsystem is provided with all VAX processors and the subsystem communicates directly with the processor logic.

TABLE 11-1 ▪ VAX Processor I/O Subsystem Allocations

		I/O Subsystem		
Processor	**UNIBUS**	**MASSBUS**	**CI Bus**	**DDI Bus**
VAX-11/725	1-standard	none	none	none
VAX-11/730	1-standard	none	none	none
VAX-11/750	1-standard 1-optional	3-optional[1]	1-optional	1-optional
VAX-11/780	1-standard 3-optional	4-optional[2]	1-optional	1-optional
VAX-11/782	1-standard 3-optional	4-optional[2]	1-optional	none
VAX-11/785	1-standard 3-optional	4-optional[2]	1-optional	1-optional

Processor	I/O Subsystem UNIBUS	MASSBUS	CI Bus	DDI Bus
VAX 8600:				
SBIA 0 (standard)	1-standard 2-optional	none	1-standard	none
SBIA 1[3] (optional)	4-optional	4-optional	2-optional[4]	4-optional[4]

Notes: (1) The number of MASSBUS subsystems is reduced to two when the optional UNIBUS subsystem is installed.

(2) The number of MASSBUS subsystems is reduced by one for each optional UNIBUS subsystem installed.

(3) The optional SBIA 1 can include up to eight I/O adapters in any combination, provided that the 13.3, Mbyte-per-second SBI aggregate bandwidth is not exceeded.

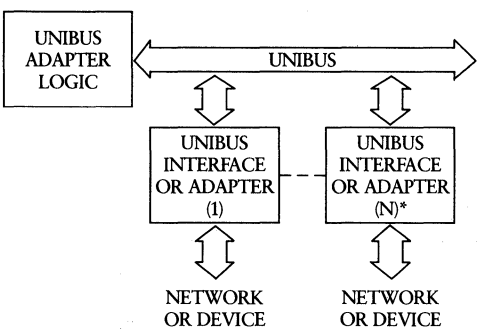
(4) Two CI780 adapters or two DR780 adapters may be installed on SBIA 1, or one CI780 adapter and one DR780 adapter may be installed.

UNIBUS Subsystem

The UNIBUS subsystem, included with all VAX processors, allows low- and medium-speed, asynchronous, bidirectional data transfers with many types of terminal, data storage, and network devices. VAX-11/725 and VAX-11/730 processors contain a single UNIBUS subsystem. The VAX-11/750 processor includes a UNIBUS subsystem, and one UNIBUS subsystem can be added by installing the DW750 UNIBUS adapter (UBA). The VAX-11/780 series of processors contains a UNIBUS subsystem and three can be added. Each additional UNIBUS requires a DW780 adapter. The VAX 8600 processor includes a UNIBUS subsystem on the first SBIA and two can be added by installing a DW780 UNIBUS adapter for each subsystem. A total of four UNIBUS subsystems can be connected to the optional second SBIA.

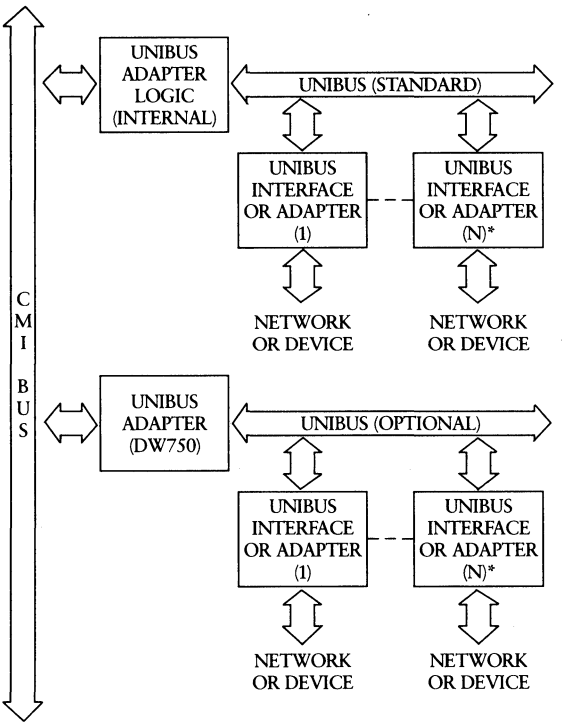
The UBA connects the processor logic to the data, address, and control lines of the UNIBUS. It performs address translation and device priority arbitration, and permits the processor to access device registers. Figures 11-1 through 11-3 show the configuration of the UNIBUS subsystem for the VAX processors. The UNIBUS offers the following features and functions:

- Permits direct memory access (DMA) transfers at high data transfer rates without CPU intervention.
- Simplifies I/O programming by allowing UNIBUS device registers to be accessed using the same addressing scheme used to access main memory.
- Allows data to be directly transferred between devices without CPU involvement.
- Supports many types of standard Digital devices and non-Digital devices.
- Permits direct-vectored hardware interrupts to service routines, thereby eliminating polling tasks of the processor.
- Contains buffered data paths to increase data throughput rates.



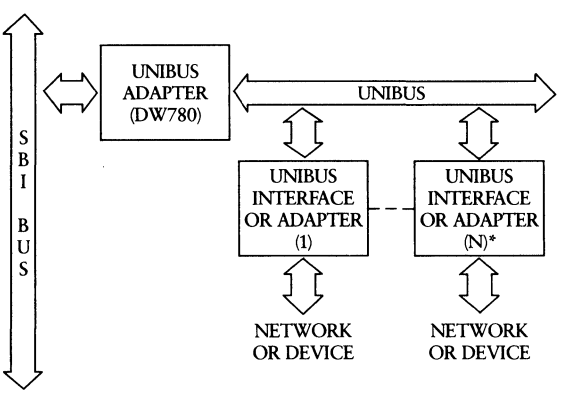
*Total number of adapters depends on adapter type and system configuration.

Figure 11-1 ▪ Small-VAX Processor UNIBUS Subsystem



*Total number of adapters depends on adapter type and system configuration.

Figure 11-2 • Medium-VAX Processor UNIBUS Subsystem



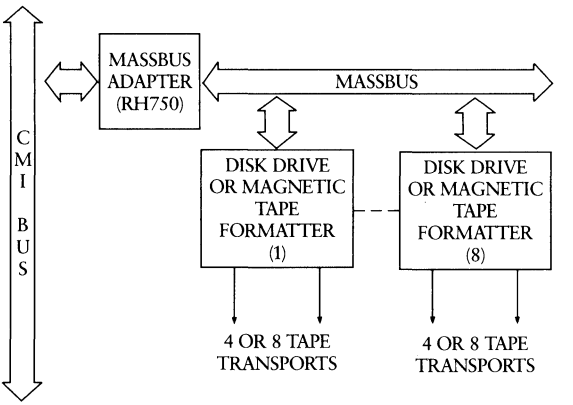
*Total number of adapters depends on adapter type and system configuration.

Figure 11-3 • Large-VAX Processor UNIBUS Subsystem

MASSBUS Subsystem

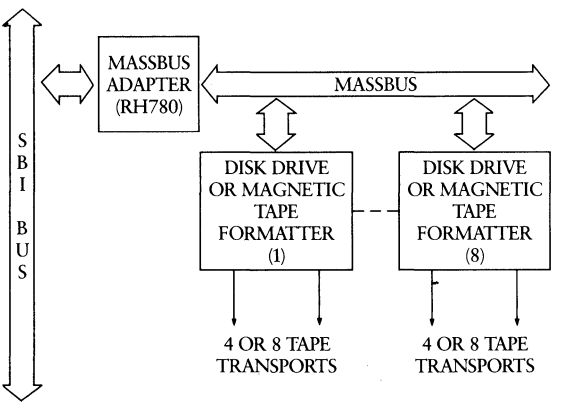
The MASSBUS subsystem, an optional I/O subsystem for medium- and large-VAX processors, permits synchronous data transfer to high-speed mass storage devices at rates of up 2 Mbytes per second. Devices that operate with the MASSBUS include large-capacity disk drives and magnetic tape devices. The MASSBUS subsystem includes diagnostic programs that allow online testing of MASSBUS devices. The RH750 MASSBUS adapter (MBA) is used with the VAX-11/750 processor and the RH780 MBA is used with large VAX processors. The MBA connects the processor logic to the MASSBUS data, address, and control lines. The MBA performs address translation and priority arbitration and allows the processor to access the registers in the devices connected to the MASSBUS. Figures 11-4 and 11-5 show the configuration of the MASSBUS subsystem for medium- and large-VAX processors. The features of the MASSBUS subsystems are as follows:

- Permits synchronous data transfers at rates of up to 1.3 Mbytes per second.
- Allows direct memory access (DMA) transfers without processor intervention.
- Contains a 32-byte silo to buffer data transferred between main memory and MASSBUS devices.
- Contains internal diagnostic features that allow online diagnosis of the MASSBUS subsystem and tape and disk drives.
- Allows MASSBUS device registers to be addressed in the same manner that main memory is addressed.
- Maps virtual addresses to physical addresses.
- Transfers interrupts from mass storage devices to the SBI.



*4 tape transports for TU77 magnetic tape formatter.
8 tape transports for TE16 magnetic tape formatter.

Figure 11-4 • Medium-VAX Processor MASSBUS Subsystem



*4 tape transports for TU77 magnetic tape formatter.
8 tape transports for TE16 magnetic tape formatter.

Figure 11-5 • Large-VAX Processor MASSBUS Subsystem

Computer Interconnect Subsystem

The Computer Interconnect (CI) subsystem is an option for medium- and large-VAX systems that permits high-speed, serial-data transfers between VAX processors and between VAX processors and intelligent mass storage subsystems. The CI subsystem consists of a CI adapter and two transmit and two receive coaxial cables. Data can be exchanged at rates of up to 70 Mbits per second. The dual-path capability ensures the dependable transfer of data and control information. The CI750 adapter is used with the VAX-11/750 processor and the CI780 adapter is used with the large-VAX processors. The CI adapter is the interface between the internal processor bus and the coaxial cables that connect to the SC008 Star Coupler unit. The SC008 unit is common connecting point for up to 16 VAX systems or mass storage subsystems. The CI adapter performs address translation and data formatting, and transposes the information during both transmission and reception. Figures 11-6 and 11-7 show the configuration of the CI subsystems, which offer the following features:

- Contains a dual-path, serial-line interface that enables high-speed and efficient information transfers.
- Easily installed to facilitate local network expansion of existing VAX systems or expansion of storage subsystems in a VAXcluster.
- Permits simultaneous operation of dual transmission paths to increase data throughput rates and the reliability of transmission.
- Eliminates control-line functions by transferring packet-oriented information.
- Contains buffered communication ports to increase data transfer rates.
- Reduces system overhead by optimizing communication and by performing error checking and bus contention functions.

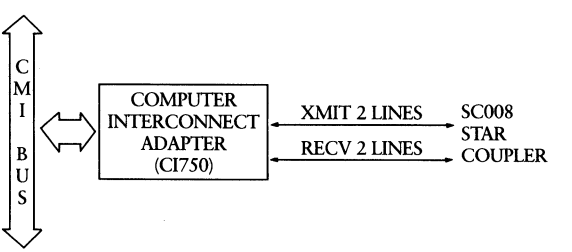


Figure 11-6 ▪ Medium-VAX Processor CI Subsystem

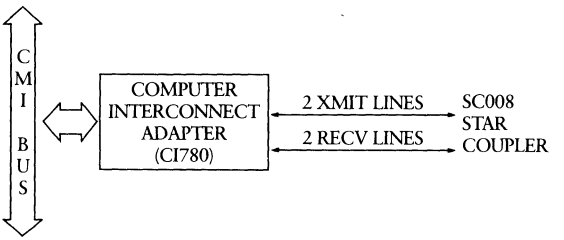


Figure 11-7 • Large-VAX Processor CI Subsystem

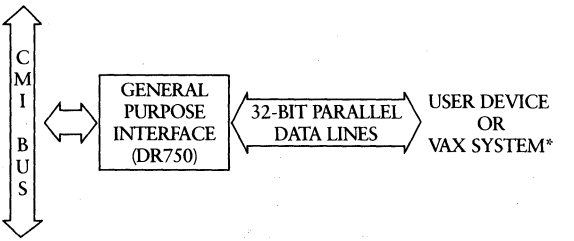
DR32 Device Interconnect Subsystem

The DR32 Device Interconnect (DDI) is a high-performance, general purpose subsystem that enables bidirectional data communication between VAX systems and user-developed devices or between VAX processors. The subsystem connects to the CMI bus of the VAX-11/750 processor and to the SBI bus of the large VAX processors. It permits continuous streams of 32-bit parallel data to be transferred to or from memory at rates of up to 6.67 Mbytes per second. The DR32 consists of a DDI adapter and bus. The DR750 DDI adapter is used with VAX-11/750 processor and the DR780 DDI adapter is used with the VAX-11/780 series processors and the VAX 8600 processor. The DDI adapter is an intelligent interface that formats data and controls data transfer. The DDI is fully supported by the VMS operating system with a simple, easy-to-use I/O driver routine and a library of high-level language support routines. The function and configuration information of the DR32 are defined in the Digital DR32 interface specification.

The DDI subsystem configuration for the VAX-11/750 processor is shown in figure 11-8 and the VAX-11/780 series processors and the VAX 8600 processor are shown in figure 11-9. The features of the DDI are as follows:

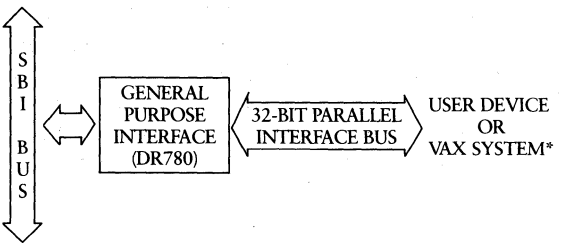
- Permits communication between VAX processors and between VAX processors and Digital- or customer-developed devices.
- Contains 32 parallel data lines for high resolution and 8 control lines for versatile device configurations.
- Allows I/O transfer rates up to 6.7 Mbytes per second.
- Permits the chaining of commands so that multiple I/O operations can be performed without software intervention.

- Allows user-generated command packets to be processed without the intervention of the operating system.
- Permits data chaining to allow continuous streams of data to be transferred to and from memory.



*VAX-11/750, VAX-11/780, VAX-11/782, VAX-11/785 or VAX 8600.

Figure 11-8 ▪ Medium-VAX Processor DDI Subsystem



*VAX-11/750, VAX-11/780, VAX-11/782, VAX-11/785 or VAX 8600.

Figure 11-9 ▪ Large-VAX Processor DDI Subsystem

I/O Subsystem Operation

Each I/O subsystem performs similar functions when operating with a VAX processor; however, the method of subsystem implementation may vary depending on the processor. The following paragraphs describe the general I/O subsystem operation, register formats, and the functional differences between processors. The operation of the CMI bus of the VAX-11/750 processor and the SBI bus of the large VAX processors is described in Chapter 9, "VAX Computer Bus Interconnects."

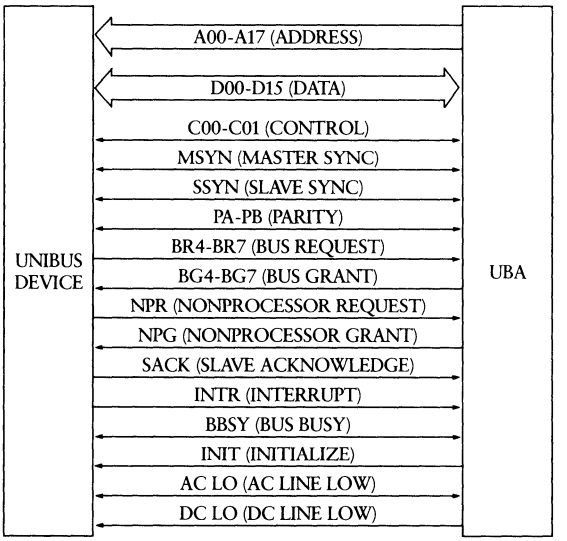


Figure 11-10 • UNIBUS Signal Lines

UNIBUS Subsystem Functions

The UNIBUS subsystem is the communication path that connects low- and medium-speed devices and networks to the VAX processors. A complete series of standard and special devices has been developed to operate with the UNIBUS. A UNIBUS subsystem can be added to an existing VAX system by installing a UBA adapter module and cable. The DR750 adapter is used with the VAX-11/750 processor and the DR780 adapter is used with the VAX-11/780 series processors and the VAX 8600 processor.

- UNIBUS SIGNAL LINES

The 56 lines of the UNIBUS transfer data, address, and control signals between the UBA and devices as shown in figure 11-10. The data information is transferred through 16 lines, and the address information is transferred through 18 address lines. The remaining lines provide status, control, and voltage monitoring functions. Table 11-2 lists the functions of the UNIBUS lines.

TABLE 11-2 ▪ UNIBUS Line Functions

Line	Function		
A00-A17	Address – Transfers memory or device register addresses. Line A00 specifies the transfer of a byte or word of data: 1 = byte and 0 = word.		
D00-D15	Data – Transfers data information.		
C00-C01	Control – Determines the data transfer operation as follows:		
	C01	**C00**	**Operation**
	0	0	*DATI* (Data in) – Transfer a word or byte from slave to master.
	0	1	*DATIP* (Data-in-pause) – Transfer a word or byte from master to slave. This is followed by either a *DATO* or *DATOB* transaction.
	1	0	*DATO* (Data out) – Transfer a word from master to slave.
	1	0	*DATOB* (Data-out-byte) – Transfer a byte from master to slave. Bit A00 specifies the byte to be written, as follows: 0 = low byte (D07-D00) and 1 = high byte (D15-D08).
MSYN	Master synchronization – Asserted by the master device to indicate that valid address and control information is on the UNIBUS.		
PA-PB	Device parity – Indicates the parity of the device information as follows: PA is not used, PB when true indicates a device parity error.		
BR7-BR4	Bus requests – Asserted by peripheral devices to request control of the bus.		
BG4-BG7	Bus grant – Asserted by UBA in response to a bus request.		
NPR	Nonprocessor request – Asserted by a device to request an NPR transfer.		
NPG	Nonprocessor grant – Asserted in response to an NPR request.		
SACK	Slave acknowledge – Asserted by a device in response to receiving a bus grant.		
BBSY	Busy – Asserted by the bus master to indicate that the bus data lines are in use.		
INIT	Initialize – Asserted by the UBA to initialize the bus devices after DC LO line is asserted.		

Line	Function
AC LO	ac line low – Asserted to indicate that the ac voltage is below the acceptable standards; initiates the UNIBUS powerfail sequence or terminates the device operation.
DC LO	dc line low – Asserted to indicate that the dc voltage is below the acceptable standards.

- UNIBUS ADAPTER OPERATION

The UNIBUS adapter (UBA) is the data path for the transfer of information between the processor and UNIBUS devices. It enables access to device data and control information by translating internal bus addresses from the processor into addresses of the UNIBUS and device registers. It also translates the UNIBUS addresses into physical memory addresses during DMA transfers, which are performed through fully buffered ports. The UBA provides priority arbitration among the UNIBUS devices that request the use of the bus. Direct-vectored interrupt capability is performed for both program and functions and for I/O device operations. The UBA also initiates a powerfail sequence when a power failure occurs or when it is directed by the program.

- COMMUNICATION AND CONTROL

A master-slave relationship exists during communication between any two devices on the UNIBUS. During a bus operation, the device in control of the bus is the bus master and the device being addressed is the slave. A processor or peripheral device can be bus master and also can access information from another device as a slave. The processor may transfer bus control to a disk drive, which can then communicate with the processor memory as a slave. When two or more devices contend for bus master, a priority scheme determines which device will obtain bus control. All control signals are interlocked so that a signal issued by the bus master must be acknowledged by a reply signal from the slave. The four types of data transactions that can be initiated between master and slave are described as follows:

- Data in (*DATI*) – Transfers a word or byte from slave to master.
- Data in, pause (*DATIP*) – Initiates a read-from-slave or write-to-slave transaction during a single bus cycle. This command is followed by a *DATO* or *DATOB* transaction to the same location to complete the sequence.

- Data out (*DATO*) – Transfers a word of data from master to slave.
- Data out, byte (*DATOB*) – Transfers a byte of data from master to slave.

- ADDRESS TRANSLATION

The UBA contains a UNIBUS map that translates device-generated addresses into physical memory addresses. It also identifies UNIBUS transfers so that other features, such as odd-byte addressing, can be appropriately used.

The 18 UNIBUS address lines can select addresses of 256 Kbytes. The UBA map is grouped into two regions, a lower address region consisting of 248 Kbytes and an upper 8-Kbyte region. The lower region is assigned to the device memory locations and the upper region to the addresses of the device control and status registers.

The address space is sectioned into 512 pages, each consisting of 512 bytes. Addresses that are contiguous in the virtual address space may be noncontiguous in the physical address space of the 512-byte boundaries. Because UNIBUS devices often use sequential addresses, the UBA breaks the sequential addresses to select the proper 512-Kbyte block. The upper 9 bits of the address are decoded in the UBA map to provide the page frame number (PFN) of the physical memory address. This information is concatenated with the word or byte location within a page that is decoded from the remaining 9 bits of the address information. The map also specifies whether the address should be incremented by one to select an odd memory location and whether the direct data path or one of the three buffered data paths should be used.

- PRIORITY ARBITRATION

Two types of data transmissions can be performed on the UNIBUS subsystem: bus request (BR) transfers and nonprocessor request (NPR) transfers. The BR interrupts the processor operation and directs the processor to perform a device routine to service the requesting device. The interrupting device sends the processor a vector address that is a location in memory containing the address of a service routine. The NPR allows the transfer of a word or byte of information from a direct memory access (DMA) device directly to memory without intervention of the processor.

The priority arbitration logic is contained in the processor and in the UBA. The priority structure, shown in figure 11-11, includes priority levels NPR and BR7 through BR4, which are assigned during the installation of the system. The NPR line is the highest priority level and the BR4 line is the

lowest. Each request line has an associated bus grant line, NPG for the NPR line and BG7 through BG4 for the BR7 through BR4 lines. These lines are distributed serially through each device that is positioned at the same level. Devices that are assigned a high priority level will be granted the bus use ahead of devices with a lower priority level. When devices are assigned the same priority level, the device that is electrically closest to the processor has precedence over more distant devices. If the processor has control of the bus and more than one device with higher priority than the processor request control, the processor will grant the bus to the device with the highest priority. If the requesting devices are of the same priority, the reply signal from the processor will be transferred through the priority line to the nearest requesting device, passing through nonrequesting devices in the process. The requesting device then takes control of the bus, executes one or more bus transactions and relinquishes the bus. The next device on the priority line will then be granted control. When all requests have been granted, the control of the bus is returned to the processor. The processor is typically assigned a lower priority than devices performing time-critical functions such as realtime process control where data loss can be critical.

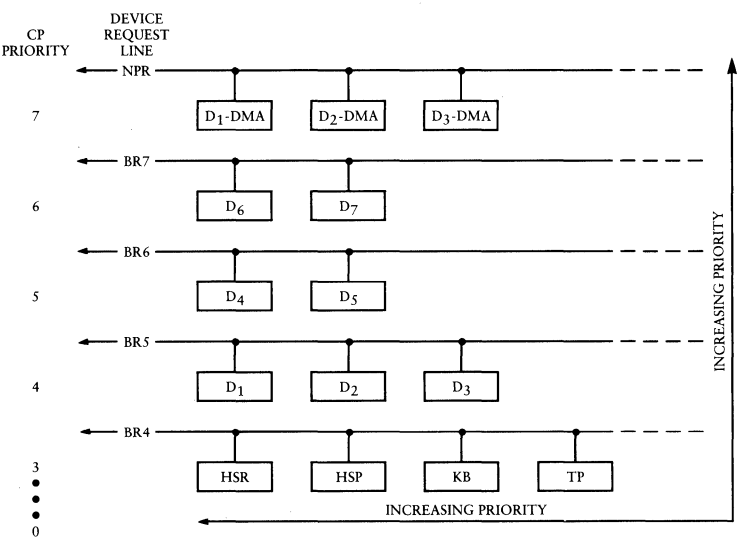


Figure 11-11 • UNIBUS Priority Assignments

Priority is assigned according to the operating speed of the device, the ease of data recovery, and the service requirements. Disk and tape storage devices are usually assigned a higher priority than line printers because of their faster transfer rates. Printers normally operate at slower speed than storage devices and the data is available for a longer period of time. The processor priority level is programmable and consists of up to 32 levels, with levels 14 through 17 (hexadecimal) corresponding to UNIBUS levels BR4 through BR7.

- DATA PATHS

A direct data path and three buffered data paths are included in the UBA. The direct data path is used for devices that interrupt the processor for each bus transaction, and the buffered data paths are used for devices capable of NPR transfers that do not require the control of the processor.

Using the direct data path, 2 bytes of data are transferred between the memory and the device during each transaction. The UBA ensures that each UNIBUS transaction results directly in a memory transaction when the data is transferred.

- The buffered data paths are used with direct memory access (DMA) transfers to optimize the use of memory and to increase the transfer speed. Each buffered data path operates as a cache and contains an address register and a 4-byte buffer register. The data is assembled into a 4-byte longword before each transaction with memory. A memory cycle is required only when the cache is not able to accept a request. If more than one NPR device is performing address transfers, the addresses that arrive at the UBA will not be sequential. During most DMA transactions, the memory locations are sequentially accessed and processor action is not necessary. Because requests are processed in a buffer before a memory access can occur, the buffered data paths improve UNIBUS access time. A memory cycle is required only when a request cannot be accepted from the cache. The three buffers ensure that the addresses are sequential when more than one NPR device is transferring information at the same time. The address received by the UBA will not be sequential.

The VMS operating system supplies standard system routines to allocate and release the buffered data paths and blocks of map registers. Standard routines also set up the allocated map registers for a transfer and convert the buffer address to a UNIBUS space address for loading into the device buffer address register. Addresses are provided in I/O space for loading the map and controlling the buffered data paths when VAX/VMS software is not used.

- DEVICE REGISTERS

  The transfer of data and status information through the UNIBUS is controlled by the device and device-control unit registers. The device registers are assigned addresses similar to the addresses assigned to memory locations and can be accessed by word-type memory-referenced instructions such as *MOVW* or *BITW* instructions. Control and status functions are assigned to the bits within the addressable register. The device service operations are controlled by setting and clearing the bits in these registers. Internal device status can be loaded into the registers and retrieved when a program instruction addresses the register. The entire register or the individual bits within a register can be read from or written to, or both read and written, depending on the register configuration. The functions of the device registers are defined in the user documentation. The functions of the UNIBUS registers are described in the sections that follow.

- INITIALIZATION AND POWER FAILURE

  The UBA controls powerfail, powerup, and initialization operations of the UNIBUS. The UNIBUS remains in a powered down state when the UBA is powered down. When the system power is applied, the UBA initiates a powerup sequence and when completed, the UNIBUS issues an interrupt request to the processor. The powerup sequence initializes the registers and functions of the UBA. When a power failure is detected, the UBA initiates a powerfail sequence resulting in an interrupt request to the processor. The UNIBUS remains in a powered down state until the power is restored and the UNIBUS powerup sequence is completed. A powerfail condition can be initiated by the program to induce the powerfail sequence when required to test the initialization procedure.

Small- and Medium-VAX Processor UNIBUS Operation

The operation of the UNIBUS is similar in the small- and medium-VAX processors. The VAX-11/725, VAX-11/730, and VAX-11/750 processors include one UNIBUS subsystem. An additional UNIBUS subsystem can be installed on the VAX-11/750 processor.

Interrupt requests are generated by devices, controllers, and memory and cause the processor to change its operation from the normal instruction execution to a service routine. The interrupting device supplies a vector address, which directs the processor to a memory location that contains the starting address of the interrupt service routine. When servicing the interrupt, the processor raises its priority to the level of the interrupting device. Interrupt requests from devices on the UNIBUS are directly addressed through the system control block (SCB) in the processor. In small-VAX processors, the address of the UNIBUS device routine is determined by adding a 200 (hexadecimal) value along with the vector value to the system

control block base (SCBB) address. In medium-VAX processors, where service routines for the first UNIBUS device are determined in the same way as for the small-VAX processors, 400 (hexadecimal) is added to the SCBB and the vector value for devices on the second UNIBUS. The device vectors assigned are the same as for PDP-11 processor UNIBUS.

During DMA transfers, the UBA performs the address translation to separate the sequential addresses transferred from the NPR device into noncontiguous 512-byte boundaries of the main memory. No restrictions are imposed on the alignment of the data in memory; however, the data words must be transferred on even addresses only. Because the UNIBUS devices can address a maximum of 256 Kbytes of memory, the address translation process of the UBA gives the devices access to the full range of physical address space. The UBA contains three buffered data paths to enable the UNIBUS devices to access the data in 4 bytes, thereby providing efficient transfers.

Processor access with a physical address of FC0000 through FFFFFF (hexadecimal) maps directly to UNIBUS addresses 0 through 777777 (octal). In the VAX-11/750 processor, the optional second UNIBUS has a physical address range of F800000 through FBFFFFF (hexadecimal). Internal devices, other than the processor, that reference the second UNIBUS address range will cause the device to receive a nonexistent memory error confirmation. When this occurs in the VAX-11/725 or VAX-11/730 processor, the device receives a timeout error.

The UBA responds to byte- or word-aligned transactions. The types of operations are:

Transaction	Operation
Read	*DATI*
Read and modify	*DATIP*
Read lock	*DATIP*
Write	*DATO* or *DATOB*
Write unlock	*DATO* or *DATOB*

The information on the UNIBUS address lines is a pointer to the UNIBUS map location, which then provides the physical memory address. Address bits A17:A09 are used to address a 512-byte by 23-bit memory location in small-VAX processors and a 512-byte by 19-bit location in the medium-VAX processors. In the small-VAX processors, the map data field is grouped

into three sections consisting of the page-frame number, the byte offset, and the valid bit. In medium-VAX processors the map data field also includes a data path number. The format of the map data fields are shown in figure 11-12.

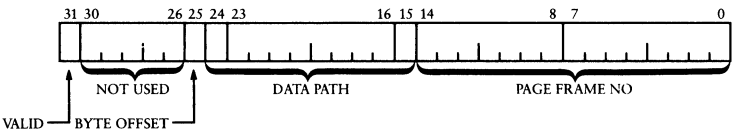


Figure 11-12 • UBA Map Data Field Format

Bit	Function
31	VALID – Set to indicate that the information from the UNIBUS map is valid.
30:26	Not used.
25	BYTE OFFSET – Set to indicate that the UNIBUS reference is an odd byte location.
24:15	DATA PATH NO (Data path number) – A binary number that selects one of the four data paths as follows: VAX-11/725 and VAX-11/730 processors: Not used. VAX-11/750 processor: 0 = direct data, 1-3 = buffered data path 1 through 3.
14:00	PAGE FRAME NO (Page frame number) – Selects the physical memory location.

In the VAX-11/750 processor, if the valid bit (bit 31) is set, the integral UNIBUS adapter will process the transaction as described. When it is cleared, the integral UNIBUS adapter ignores the UNIBUS requests. The valid bit is cleared for map entries that correspond to sections of UNIBUS address space containing slave devices that are expected to respond to transactions originating on the UNIBUS. Transactions from the CPU that result in a UNIBUS transaction are always ignored by the integral UNIBUS adapter and can never wrap back through the UNIBUS adapter to memory.

If the byte offset (bit 25) is set, the address supplied by the DMA device is incremented by one to allow devices that generate only even-byte addresses to access buffers on odd-byte boundaries. A transaction that causes a word to cross page boundaries because the offset bit is set must have the same data path number, offset bit, and valid bit in both map entries. If the transaction is a *DATI*, *DATIP*, or *DATO* and the two bytes of UNIBUS data cross a longword boundary, two memory cycles will occur.

The data path numbers (bits 22:21) select the direct data path for the BR requests and the buffered data paths for the DMA transfers. A data path number of 0 is the direct data path and numbers from 1 to 3 select buffered data paths 1 to 3 respectively. Each of the three data paths consists of 4 bytes of data storage, a 16-bit address, and 5 flag bits. These registers and flags are not accessible to the software.

UNIBUS transactions to ascending or descending sequential addresses or to nonsequential addresses require only one data memory transfer for every two UNIBUS transfers. During UNIBUS-initiated transactions that cause a memory read or write cycle to occur, the map PFN is concatenated with UNIBUS address (bits A8:A0) to form a 24-bit physical address. Figure 11-13 shows this address translation process.

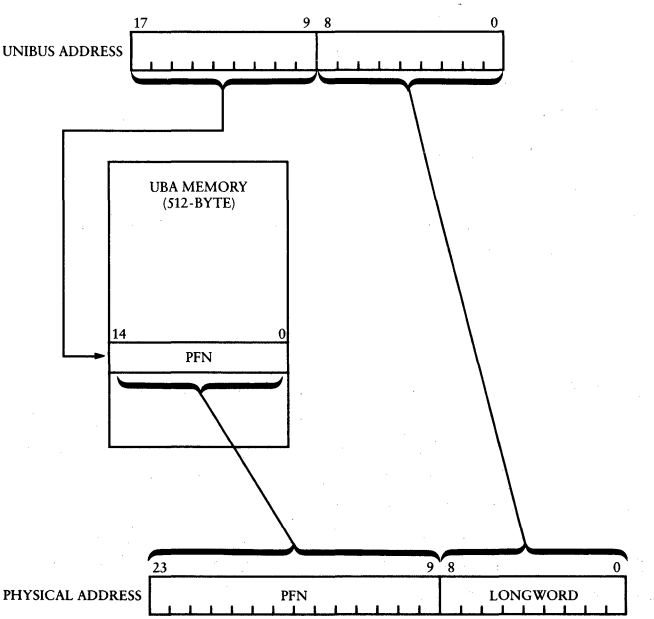


Figure 11-13 ▪ UNIBUS-to-Physical-Address Translation

The map is accessible for reading and writing. Each entry uses a longword address of from F30800 to F30FFC (hexadecimal) for the first UNIBUS and F32800 to F32FFC (hexadecimal) for the optional second UNIBUS. A longword must be sent to the map address during a write transaction or the contents of the written map location will be unpredictable.

Buffered Data Path Operation–The buffered data path can be loaded from the UNIBUS or from the memory and its contents can be transferred to the UNIBUS or memory. One or two bytes at a time can be loaded into the buffer from the UNIBUS; four bytes at a time are loaded from memory.

Five flags in the data path indicate the status of the data buffer. If the CD flag is set, the buffer has 4 bytes of data from memory and flags BF3 through BF0 are cleared. If the CD flag is cleared, flags BF3 through BF0 indicate which bytes in the data buffer have valid UNIBUS data. If the buffer is empty, then all flags are cleared.

Each buffered data path contains a 16-bit address register that can be loaded from UNIBUS addresses lines (A17:A2). The stored address is compared with the address on the UNIBUS for the same value. The address held in the register is the UNIBUS longword corresponding to the data in the data buffer. The following operations are described for each transaction:

- *DATI* and *DATIP*–The UBA performs similar operations during the *DATI* and *DATIP* transactions, as determined by the contents of the data and address buffers. These transactions allow a device to use a buffered data path to perform a sequence of transfers in either direction using any address sequence. A device that performs repeated transfers within the same longword, however, will not cause memory cycles to occur. If the buffer is empty or contains memory data but the address in the address register is not the same as the UNIBUS address, the following transactions occur: The UBA reads the memory data and transfers it to the buffer and to the UNIBUS, loads the UNIBUS address into the address register, sets the flags to indicate that data is in the buffer, and transfers the data to the requesting UNIBUS device. If the buffer contains data, the UBA first writes the stored data into memory at the location indicated by the stored address and the byte flags as the byte mask. If the buffer contains data and if the address in the address register is the same as the UNIBUS address, then data is transferred to the UNIBUS.

- *DATI* and *DATIP* with byte offset–If the map specifies a byte offset, the operation depends on whether the transaction crosses a longword boundary. If the boundary has not been crossed, then the response is identical to that described for *DATIP*. If a longword boundary has been crossed, the UBA performs one memory read operation at the given UNIBUS address and a second memory read operation at the same

address but incremented by two. The two memory reads are assembled to form the UNIBUS data word. The data buffer and address register hold the information from the second read at the end of the transaction.

- *DATO* or *DATOB* – The contents of the buffer determine the UBA operation. Whether the buffer is empty or contains data, the UBA transfers the UNIBUS data to the data buffer, loads the UNIBUS address into the address register, sets the appropriate status flags to indicate the amount of data in the buffer, and informs the UNIBUS device when the transaction is completed. If the buffer has UNIBUS data and the address in the address register is the same as the UNIBUS address, then the operation depends on whether the UNIBUS data, combined with the data in the buffer, form a longword. If a longword does not exist, the same operation occurs as if the buffer were empty or had memory data. If the data forms a longword, the data will be written into memory and the data path flags will be cleared to indicate that the buffer is empty. If the buffer has UNIBUS data and the addresses are not the same, then the data is written into memory and the data path flags are cleared to indicate that the buffer is empty. The UBA will then perform the same operations as it would with the buffer empty.

- *DATO* with byte offset – If this transaction crosses a longword boundary, the UBA performs two one-byte writes. If it does not cross a boundary, the UBA performs the same operation as with a *DATO* or *DATOB* operation.

- *DATOB* with byte offset – If this transaction does not cross a longword boundary, the same transaction is performed as with the *DATO* or *DATOB* and the address is incremented by one. If a longword boundary is crossed, a *DATOB* operation is performed in the next longword and the same addresses are forced to become different.

Control and Status Registers – Small-VAX processors have a single data path for the UNIBUS transfer operations. This data path has an associated read-only control/status register (CSR2) that provides pertinent information to the operating system to ensure the satisfactory transfer of data. Figure 11-14 shows the format of information in the register.

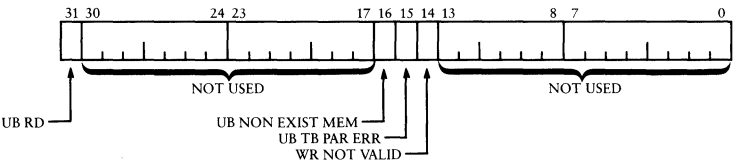


Figure 11-14 • Small-VAX Processor UNIBUS CSR2 Format

Bit	Function
31	UB RD (UNIBUS read) – Set to indicate that an uncorrectable data error has occurred during a UNIBUS read to memory.
30:17	Not used.
16	UB NON EXIST MEM (UNIBUS nonexistent memory) – Set if a UNIBUS request references a nonexistent memory array.
15	UB TB PAR ERR (UNIBUS/memory transaction parity error) – Set to indicate that the UNIBUS/memory transaction resulted in a parity error.
14	WR NOT VALID (Write not valid) – Set during a two-cycle operation if an attempt is made to write to a page that does not have a valid entry in the translation buffer.
13:00	Not used.

Medium-VAX processors contain three separate control/status registers, one for each buffered data path. These registers are used by the software when intervention with the buffered data paths is required. When a device and memory have completed a series of transactions, data may be left in a data buffer if the transfer did not end on an even longword boundary. The corresponding UNIBUS address of the data will be contained in the address register. If the contents of the map are changed before the data is removed, the association between address and data in the buffer will not be correct. The software must therefore initiate the action to write this data to memory and clear the buffer. Figure 11-15 shows the format of control/status register and the bit assignments.

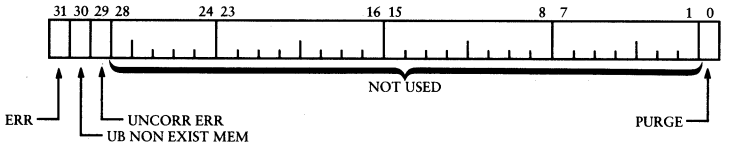


Figure 11-15 ▪ Buffered Data Path Control/Status Register Format

Bit	Function
31	ERR (Error) – Set if either the UB NON EXIST MEM (bit 30) or the UNCORR ERR (bit 29) are set.
30	UB NON EXIST MEM (UNIBUS nonexistent memory) – Set when the UNIBUS references a nonexistent memory location. No response is received on the UNIBUS and all future UNIBUS transactions through the buffered data paths are ignored until the bit is cleared by writing a one to it.
29	UNCORR ERR (Uncorrectable error) – Set when an uncorrectable error status is received from memory. The parity error (PB) line is asserted and the data is sent back to the UNIBUS device on the first read from that location. This bit is not asserted on subsequent read operations from this buffered data path and can be cleared by writing a one to this bit.
28:1	Not used.
0	PURGE – A read-write bit set to perform a transaction that depends on the contents of the buffer. If the buffer contains UNIBUS data, the data is written into memory and the flags indicate that the buffer is empty. If the buffer contains memory data, then the flags indicate that the buffer is empty. This bit is monitored by software to indicate when the memory write operation is completed.

Large VAX Processor UNIBUS Subsystem

The UNIBUS subsystems of the VAX-11/780 series processors and the VAX 8600 processor connect to the synchronous backplane interconnect (SBI) bus through the DW780 UNIBUS adapter (UBA). Each VAX-11/780 series processor includes one UNIBUS subsystem and can accommodate three more. The VAX 8600 processor can be installed with two SBI buses, one supplied with the processor (SBIA 0) and one SBI bus (SBIA 1) available as an option. One UNIBUS subsystem is supplied with SBIA 0 and two can be added. Up to four UNIBUS subsystems can be installed on SBIA 1. Each UNIBUS subsystem includes a UBA interface connected between the asynchronous UNIBUS and the synchronous SBI bus. The UBA provides access to UNIBUS device registers from the SBI, translates UNIBUS addresses to SBI addresses during DMA transfers to the processor memory, and performs priority arbitration among the UNIBUS devices. It includes a data transfer path to enable devices to access random SBI memory addresses and to permit high-speed data transfers to be performed to consecutive memory addresses from UNIBUS devices. Detailed information on the operation of the SBI is contained in chapter 9.

VAX system hardware supports a UNIBUS adapter in one of four physical address spaces. In the VAX 8600 processor, four physical address spaces are provided for the standard SBI bus and four for the optional SBI bus. The UBA maintains two independent spaces within the SBI I/O address space. The first area of addressable space is reserved for the internal registers of a nexus such as a memory controller, MBA, UBA, or CI adapter. Each register address space of each nexus occupies 8 Kbytes (16 pages each with 512 bytes) and contains the control and status registers of the UBA, registers required for UNIBUS interrupt requests, and registers required for mapping the UNIBUS device transfers to SBI address space. The second space is the UNIBUS address space associated with the UBA. It occupies a total of 256 Kbytes (512 pages each with 512 bytes). Figure 11-16 shows the assignments of the SBI I/O address space.

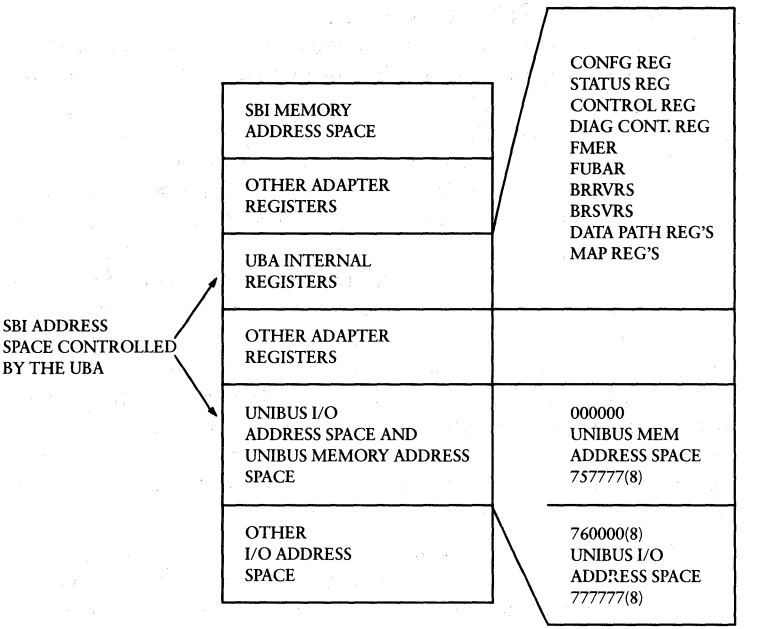


Figure 11-16 • SBI I/O Address Space Assignments

- SBI-TO-UNIBUS TRANSFERS

UNIBUS address space consists of 248 Kbytes of physical memory space and 8 Kbytes of device register space and is part of the SBI I/O address space. The UBA translates SBI commands and address information into UNIBUS commands and addresses and enables the software to read and write to UNIBUS device registers using word-type memory reference instructions such as the *MOVW* or *BITW* instructions.

Device registers are assigned I/O addresses within the UNIBUS space from 760000 to 777777 (octal) or from 201XE000 to 201XFFFF (hexadecimal). The hexadecimal digit 3, 7, B, or F is substituted for the X value within the physical address, depending upon which one of the four address spaces is assigned to the UBA. During processor-initiated transfers, the SBI translates the command into the following functions. The UBA becomes the highest priority UNIBUS NPR device.

Transaction	Function
DATI	Read masked word or byte from device.
DATO or *DATOB*	Write masked word or byte to device.
DATIP then *DATO* or *DATOB*	Interlock read-masked then interlocked write-masked.

- SBI-TO-UNIBUS ADDRESS AND FUNCTION TRANSLATION

Each SBI longword address is composed of two 16-bit UNIBUS word addresses. In addition to decoding the SBI address, the SBI function and byte mask is decoded to determine if a word or byte is to be accessed. This translation of the SBI mask and function values into UNIBUS control values is performed by the UNIBUS control and byte address encoder, as shown in figure 11-17.

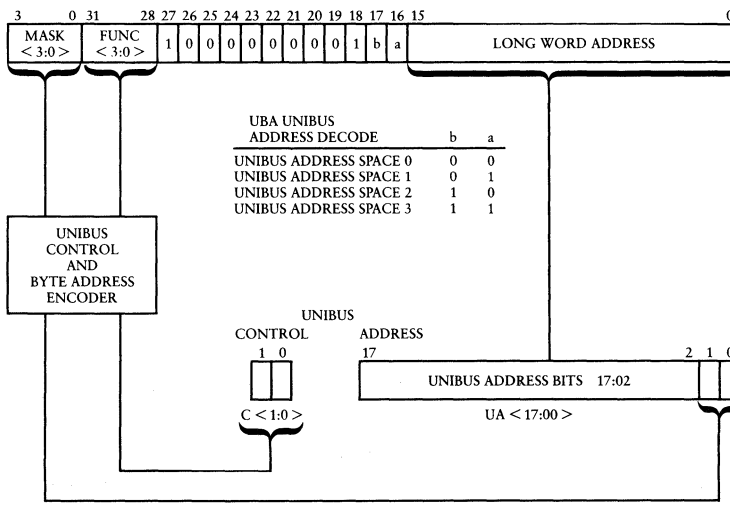


Figure 11-17 • SBI-to-UNIBUS Control Address Translation

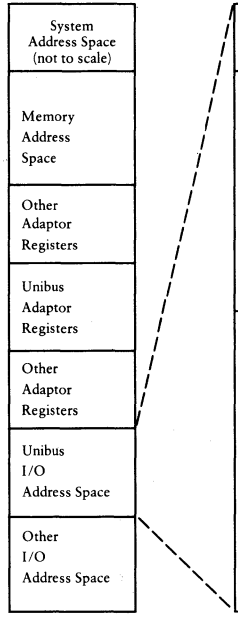
The SBI mask and function fields are translated into UNIBUS control and address fields. Table 11-3 shows the UNIBUS control values translated from the SBI function and mask values. Only the combinations shown on the table are valid; other values will cause an error confirmation. The UNIBUS address space will respond only to word or byte SBI references. Extended transfers cannot occur to either the UNIBUS address space or to the UNIBUS adapter registers.

TABLE 11-3 • SBI-Function/Mask-to-UNIBUS Translation

SBI		UNIBUS	
Function <3:0>	Mask 3 2 1 0	Control C<1:0>	Address UA<1:0>
Read Mask	0 0 0 1	DATI	0 0
	0 0 1 1	DATI	0 0
	0 0 1 0	DATI	0 0
	0 1 0 0	DATI	1 0
	1 1 0 0	DATI	1 0
	1 0 0 0	DATI	1 0
Write Mask	0 0 0 1	DATOB	0 0
	0 0 1 0	DATOB	0 1
	0 1 0 0	DATOB	1 0
	1 0 0 0	DATOB	1 1
	0 0 1 1	DATO	0 0
	1 1 0 0	DATO	1 0
Interlock Read Mask (Sets Interlock Flip Flop for DATIP-DATO Sequence)	0 0 0 1	DATIP	0 0
	0 0 1 0	DATIP	0 0
	0 1 0 0	DATIP	1 0
	1 0 0 0	DATIP	1 0
	0 0 1 1	DATIP	0 0
	1 1 0 0	DATIP	1 0
Interlock Write Mask	0 0 0 1	DATOB	0 0
	0 0 1 0	DATOB	0 1
	0 1 0 0	DATOB	1 0
	1 0 0 0	DATOB	1 1
	0 0 1 1	DATO	0 0
	1 1 0 0	DATO	1 0

When the software initiates a read or write transaction to a UNIBUS device register, the UBA recognizes that the address is within the UNIBUS address space and transfers the lower 16 SBI address bits through to the UNIBUS as address bits A17:A2. The UBA generates UNIBUS address bits A1:A0 and control bits C1:C0 by decoding the SBI mask and function bits. Table 11-4 shows the relationship of the UNIBUS space controlled by the UBA to the SBI address space.

TABLE 11-4 • UNIBUS and SBI Address Space



30 bit Physical Byte Address (Hex)				18 bit Unibus Address Space (Hex)				18 bit Unibus Address Space (Octal)				
20100002	20100000			00002	00000			000002	000000			
20100006	20100004			00006	00004			000006	000004			Unibus
2010000A	20100008			0000A	00008			000012	000010			Memory
2010000E	2010000C			0000E	0000C			000016	000014			Address
20100002	20100010			00012	00010			000022	000020			Space
.	.			.	.			.	.			Reserved
.	.			.	.			.	.			For
.	.			.	.			.	.			Expansion
2013DFF6	2013DFF4			3DFF6	3DFF4			757766	757764			(496 pages)
2013DFFA	2013DFF8			3DFFA	3DFF8			757772	757770			1 page = 512 bytes
2013DFFE	2013DFFC			3DFFE	3DFFC			757776	757774			
2013E002	2013E000			3E002	3E000			760002	760000			Unibus
2013E006	2013E004			3E006	3E004			760006	760004			I/O
2013E00A	2013E008			3E00A	3E008			760012	760010			Address
.	.			.	.			.	.			Space
.	.			.	.			.	.			
.	.			.	.			.	.			
2013E013	2013E012	2013E011	2013E010	3E013	3E012	3E011	3E010	760023	760022	760021	760020	example for DATOB
.	.			.	.			.	.			
.	.			.	.			.	.			
.	.			.	.			.	.			Upper 4 K (10) 16 bit words
.	.			.	.			.	.			
2013FFF6	2013FFF4			3FFF6	3FFF4			777766	777764			
2013FFFA	2013FFF8			3FFFA	3FFF8			777772	777770			
2013FFFE	2013FFFC			3FFFE	3FFFC			777776	777774			

Note: These addresses refer to UBAO.

During a read sequence to the UNIBUS address space, if data is received from the device with a device parity error, then the data will be sent to the SBI as a read data substitute. If an access is made to the UNIBUS address space and the transfer is not completed on the UNIBUS because of a nonexistent device, the following events will occur:

- All zeros will be sent as data for a read transfer.
- UNIBUS address (bits A17:A2) will be stored in the failed UNIBUS address register (FUBAR).
- The bit indicating the cause of the failure (UBA select timeout or slave sync timeout) will be set in the UBA control/status register. During a write transfer to the UNIBUS, the error bit is set after the command is issued and acknowledged by the UBA.

- UNIBUS-TO-SBI ADDRESS TRANSLATION

UNIBUS-initiated transfers to UNIBUS device addresses are mapped by the UBA-to-SBI addresses on a page-by-page basis. This allows data to be transferred to noncontiguous pages of SBI memory. The SBI uses a 30-bit address and a 32-bit data path, and the UNIBUS uses an 18-bit address and a 16-bit data path. The SBI operates synchronously and supports as many as 16 nexus nodes. The UNIBUS operates asynchronously and supports many devices.

The UBA accepts hardware-generated interrupts and DMA data transfers from the UNIBUS. The input transfer from a terminal is an interrupt process in which the terminal interface initiates an interrupt sequence. The interrupt service routine for the terminal driver will accept and process the data that results from the terminal input sequence. Some terminal interfaces are designed for DMA transfers, with the interface controlling the input DMA process. This process is an indirect memory transfer. Once the process is initiated by the software, a disk drive can transfer data directly to or from SBI memory through the UBA without processor intervention. The DMA transfer may be a random access of noncontiguous addresses or a sequential access of sequentially increasing addresses. The UNIBUS adapter can channel data through any one of 16 data paths for UNIBUS devices performing DMA transfers. The UBA provides a direct data path to allow UNIBUS transfers to random SBI addresses. Each UNIBUS transfer through the direct data path is mapped directly to an SBI transfer, allowing the transfer of only one word of information during an SBI cycle. The UBA provides 15 buffered data paths, each of which allows a sequential access to the memory locations from a device on the UNIBUS. Each path stores data for the UNIBUS so that four UNIBUS transfers are performed for each SBI transfer. The UBA can support high-speed, DMA block-transfer devices through the buffered data paths and allows a UNIBUS device to operate on random longword-aligned 32-bit data.

- UNIBUS-TO-SBI ADDRESS TRANSLATION

In the VAX-11/780 series of processors, DMA transfers from the UBA to main memory are through the memory controllers connected to the SBI. In the VAX 8600 processor, the DMA transfers to memory are through the A bus that connects to the Mbox and communicates with the memory. The UBA translates UNIBUS memory addresses into SBI addresses and UNIBUS memory page addresses into SBI page addresses through an address translation map containing 496 hardware map registers. Each map register is assigned an SBI longword address, and each contains the SBI page address and the data path required to transfer data between the UNIBUS and the SBI.

The UBA is mapped to an SBI address in three sections: the SBI page address, with one page equal to 512 bytes; the longword within an SBI page, with one longword equaling four bytes; and the word or byte within a longword. As shown in figure 11-18, the UNIBUS-to-SBI page map translates UNIBUS memory page addresses to SBI page addresses. The map allows the transfer of data to noncontiguous pages of SBI memory and translates the UNIBUS address (bits A17:A9) to the SBI page address (bits B27:B7).

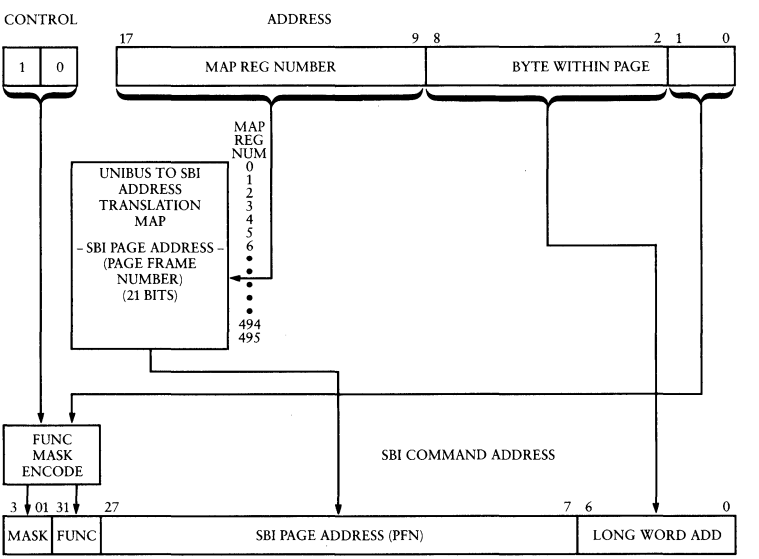


Figure 11-18 • UNIBUS-to-SBI Address Translation

The 496 map registers map the entire UNIBUS memory address space at the same time, and each map register corresponds to the UNIBUS page that is to be mapped. The map registers can be accessed by the software as part of the SBI I/O address space. UNIBUS address bits A8:A2 select the longword within a page and become SBI address bits B6:B0. These 7 bits are concatenated with the mapped SBI page address to form the 28-bit SBI address.

The two low-order UNIBUS address bits (A1:A0) and the two control bits (C1 and C0) determine the SBI function bits (F1 and F0) and the byte mask (M3:M0).

The mask field points to either one or two bytes within the longword address. The function field selects either a read or write operation and includes the associated qualifier. The translation values are listed in table 11-5.

TABLE 11-5 • UNIBUS and SBI Field Translation Values

UNIBUS Field Control	Address A1	Address A0	SBI Field Function	Mask M3	Mask M2	Mask M1	Mask M0
DATI	0	0	Read mask	0	0	1	1
	1	0		1	1	0	0
DATO	0	0	Write mask	0	0	1	1
	1	0		1	1	0	0
DATOB	0	0	Write mask	0	0	0	1
	0	1		0	0	1	0
	1	0		0	1	0	0
	1	1		1	0	0	0
*DATIP**	0	0	Interlocked	0	0	1	1
	1	0	read mask	1	1	0	0
DATO	0	0	Interlocked	0	0	1	1
	1	0	write mask	1	1	0	0
DATOB	0	0	Interlocked	0	0	0	1
	0	1	write mask	0	0	1	0
	1	0		0	1	0	0
	1	1		1	0	0	0

* DATIP is followed by a DATO or DATOB

- UBA DATA TRANSFER PATHS

Data is transferred between the UNIBUS and the SBI through the direct data path and 15 buffered data paths of the UNIBUS adapter. The data path to be used by a particular device is assigned by the software when the map registers are defined. DP0 is the direct data path and DP1 through DP15 are buffered data paths 1 through 15. One or more transferring devices can be assigned to DP0; however, only one transferring device can be assigned to a buffered data path at one time. During a DMA transfer, if the UNIBUS address points to an invalid map register or to a map register that has a parity error within the high-order 16 bits, the UNIBUS transfer will be aborted and the bit indicating the condition will be set in the UBA control/status register. For this implementation, the low-order 16 bits of the map register are accessed only when an SBI transfer is required and only at that time is parity checked on the low 16 bits of the map register.

The direct data path translates each UNIBUS data transaction directly into an SBI function for each UNIBUS word or byte transfer, thereby transferring data between SBI memory and a UNIBUS device in 16-bit quantities. The direct data path will respond to all UNIBUS data transactions. The UNIBUS transfer is complete when the SBI transfer is completed. The SBI address, function, and byte mask are mapped directly from the UNIBUS address and control lines and from the state of an internal interlock during a *DATIP* or *DATO* sequence. The direct data path is easy to program because only the map registers are accessed when initiating a UNIBUS transfer. The DDP is used by devices executing an interlock sequence (*DATIP-DATO/DATOB*) to the SBI, by devices transferring to consecutive increasing addresses, or by devices that mix read and write functions. The maximum data throughput rate is approximately 425 Kwords per second for write operations and 316 Kwords per second for read operations; however, the rates will decrease when more than one transfer is in process simultaneously. Table 11-6 lists the transfer operations that can be performed through the direct data path.

TABLE 11-6 • Direct Data Path Transfer Functions

UNIBUS Function	Transfer Direction	SBI Function
DATI	UBA to device	Read mask (16 bits)
DATO or *DATOB*	Device to UBA	Write mask (8 or 16 bits)
DATIP and *DATO* or *DATOB*	UBA to device, then device to UBA write-mask	Interlock read-mask then interlock

The buffered data paths enable the fast and sequential block transfers between physical memory and the UNIBUS devices through the SBI. Each of the 15 buffered data paths stores the data from a device as words or bytes and then transfers 64 bits to SBI memory, except for the VAX 8600 processor, which does not use the SBI memory. During output transfers, one quadword from memory is stored and then transferred to the device as a word or byte. The buffered data paths will respond to all UNIBUS data transactions except the *DATIP* function. These data paths also allow a UNIBUS device to operate on random 32-bit longword-aligned data and enable word-aligned block transfer devices to begin and end on an odd byte of SBI memory. A UNIBUS device can operate on random longword-aligned 32-bit data from SBI memory so that all 32 bits of the longword are read or written at the same time. The buffered data path is selected by software when the map registers are formatted. Only one active transfer is assigned to the data path at one time.

A block is greater than or equal to one byte, and transfers within a block must be to consecutive increasing addresses. All transfers within a block must be of the same function type, such as a memory read (*DATI*) or a memory write (*DATO* or *DATOB*).

- DATA TRANSFERS TO MEMORY

After the device addresses the last byte or word of a physical quadword, the UBA completes the data cycle and performs an extended write operation to transfer the stored bytes of data. The SBI transfer is completed before additional transfers can be performed. The control/status register indicates to the program whether valid data is contained in the buffer. The format of the information in a buffer data path is shown in figure 11-19. Four 16-bit data words are transferred. The last transfer initiates an extended write transfer of all 64 bits to memory. The buffer stores the UNIBUS addresses of the data it contains so that the remaining bytes can be transferred to memory at the end of a block transfer. The buffer also records the type of function and the state of each stored byte in the data buffer. This information is transmitted as the SBI mask bits during write cycle to allow only the correct bytes to be written into memory.

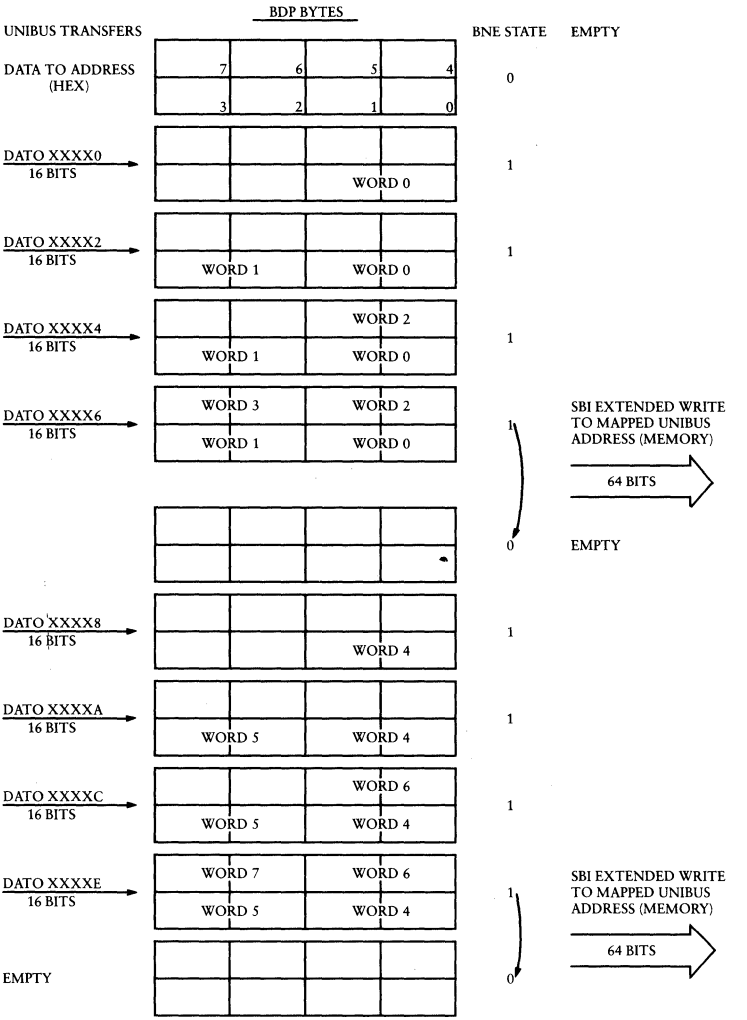


Figure 11-19 • UNIBUS Data Transfers to Memory

- DATA TRANSFERS FROM MEMORY

During data transfers to memory, the UBA logic monitors the state of the data buffers. If the buffers are empty, an extend read operation to memory will be initiated. The data for the current cycle will be transferred and the cycle completed, the remaining bytes will be stored in the buffers, and the "buffer not empty" condition will be indicated. If the data for the current cycle is available in the data buffers, then the buffer will pass the data to the UNIBUS and complete the cycle. The buffer prefetches the next quadword of data using an extended read transfer after each UNIBUS access to the last word of a quadword-aligned group. The "buffer not empty" condition is cleared before the prefetch action and set when the data is read. Since the prefetch action would allow a page boundary to be crossed into nonexistent memory, a timeout condition could occur. This is prevented by the software allocation of an additional map register following a block. The map register therefore must be invalidated. When a page boundary is crossed to the invalid map register, the prefetch will be aborted before a timeout occurs. The UBA does not record any UBA or SBI errors that occur during the prefetch operations. If an error occurs, the prefetch will be aborted and the buffer will remain empty. If the same buffer is again accessed by the device, then the read operation will be initiated and any errors that occur will be logged. Figure 11-20 shows the buffered data path transfer from memory to the UNIBUS.

Devices that begin transfers on word boundaries and transfer an integral number of words can begin and end a block transfer at an odd byte of SBI memory. The byte offset bit of the map registers must be set by software during the devices transfer. This effectively increases the SBI memory address by one byte. The data on the UNIBUS is the byte or word indicated by the UNIBUS address. The data on the SBI is increased by one byte. The UBA will distribute the data and adjust the SBI address and byte mask so that the data will be transferred to the correct memory location. The device does not intervene in this process. Figure 11-21 shows the relative position of data transferred between a device and SBI memory.

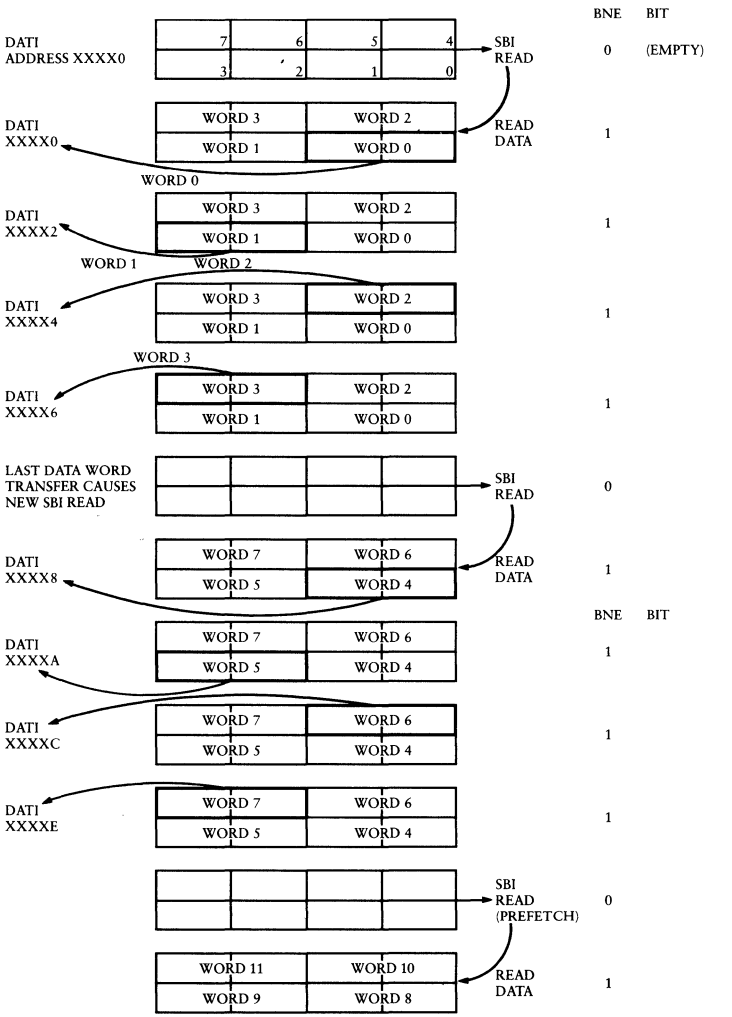
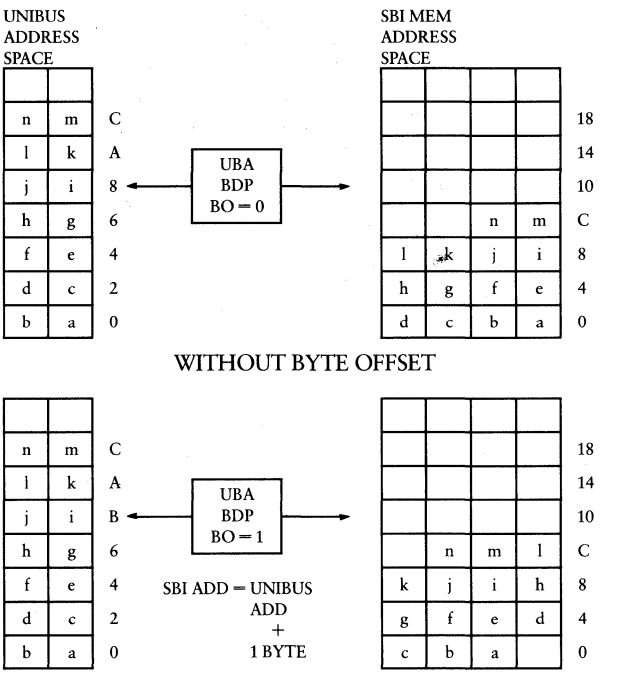


Figure 11-20 ▪ UNIBUS Data Transfers from Memory



Note: Each box represents one byte of data.

Figure 11-21 ▪ Relative Position of UNIBUS and SBI Data

The buffered data paths are cleared of information and the registers are initialized by the software at the completion of a device transfer to memory. Bytes of data that remain in the buffer will be transferred to memory. The UBA will then indicate that the buffer is empty. If an error occurs during this transfer, the error will be indicated in the data path register indicating an unsuccessful transfer. The software must clear this bit before the buffer is again available for transfer. If no data remains in the buffer at the end of the operation, then the buffer is left in its initialized state. During read operations from memory the UBA initializes the buffers by indicating that the buffers are full.

- LONGWORD-ALIGNED 32-BIT RANDOM ACCESS MODE

Data can be transferred between a device and memory using longword-aligned 32-bit quantities in a random access mode. A buffered data path selected by the map register is used for this mode. A UNIBUS device first transfers the low-order word of the longword, then the high-order word. This is performed with a read from memory (*DATI*), write to memory (*DATO*), or a read/write (*DATI/DATO*) operation. The UBA operation for this mode is determined by the transaction and address received from the UNIBUS and the state of the buffer. The *DATIP* function code is not valid for these transfers. No prefetch operation will be performed when this mode is enabled. The initialization operation at the completion of the transfer is not performed when the device transfers both words of the longword in the specified order. Maximum throughput in this mode is approximately 1.7 Mbytes per second in the sequence shown in figure 11-22.

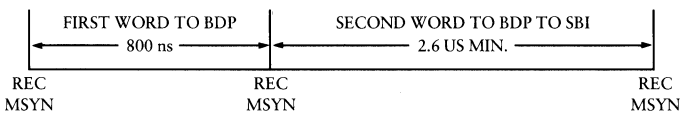


Figure 11-22 • Random Access Mode Throughput

The SBI read operations will occur when a *DATI* operation is received and the buffered data path is empty. If the buffer contains data, the data will be returned to the device. The "buffer not empty" condition will be reported at the completion of a transfer of the low-order word and cleared after the high-order word is transferred.

The "buffer not empty" condition is indicated during a *DATO* or *DATOB* operation. The data from the device is stored in the buffer and the byte mask bit is set within the data path register to indicate the bytes or words that have been written by the device. The *DATO* operation is performed on the high-order word or when a *DATOB* operation occurs to the high-order byte. The bytes or words that have the byte mask bits set are written into main memory. The "buffer not empty" bit is cleared after a SBI write operation.

Table 11-7 lists the UBA operations for each UNIBUS transaction.

TABLE 11-7 • UBA Operations for UNIBUS Transactions

UNIBUS Transaction	Address A0	Address A1	UBA Operation	BNE State Before	BNE State After
DATI	0	X	SBI read, return low word, store data	0	1
	1	X	SBI read, return high word	0	0
	0	X	Return low word	1	1
	1	X	Return high word	1	0
DATO	0	X	Store low word	0	1
	1	X	Store high word, SBI write	0	0
	0	X	Store low word	1	1
	1	X	Store high word, SBI write	1	0
DATOB	0	0	Store byte 0	0	1
	0	1	Store byte 1	0	1
	1	0	Store byte 2	0	1
	1	1	Store byte 3, SBI write	0	0
	0	0	Store byte 0	1	1
	0	1	Store byte 1	1	1
	1	0	Store byte 2	1	1
	1	1	Store byte 3, SBI write	1	0
DATIP	X	X	UBA does not respond (nonexistent memory to UNIBUS device); no change	0	0
	X	X	UBA does not respond (nonexistent memory to UNIBUS device); no change	0	0

Figure 11-23 shows the map register bit assignments required to program the UBA for the longword-aligned random access mode.

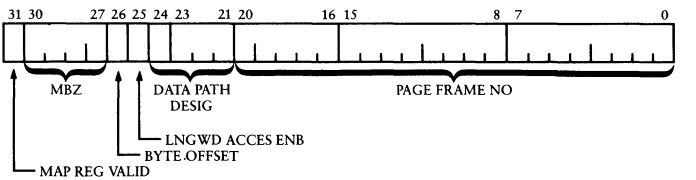


Figure 11-23 • Longword-aligned Map Register Format

Bit	Function
31	MAP REG VALID (Map register valid) – Set to indicate that the map register information is valid.
30:27	MBZ (Must be zero).
26	LNGWD ACCES ENB (Longword access enable) – Set to initiate the longword-aligned random access operation.
25	BYTE OFFSET – Cleared to indicate that the byte is not offset.
24:21	DATA PATH DESIG (Data path designators) – A binary value used to select the buffered data path, as indicated: 0 = direct data path 1-15 = buffered data paths 1 through 15
20:0	PAGE FRAME NO (Page frame number) – SBI page address.

• INTERRUPT REQUESTS

The SBI interrupts requests are initiated from the UNIBUS subsystem by a device or by the UBA on one of the four UNIBUS bus request (BR) lines. The UBA level is selected by a backplane jumper lead. The UBA contains one request sublevel and requires 4 of the 64 possible SBI interrupt vectors, one for each of the four required levels. Each of the four vectors selects a UBA service routine corresponding to an interrupt request level. Each UBA service routine reads and tests the BR receive vector register (BRRVR 7-4) that corresponds to the level of interrupt (7-4). The UBA service routine determines whether the interrupt was generated from within the UBA status register, from the UNIBUS device, or from both. The UBA service routine services the interrupt as specified by the contents of the BRRVR. Figure 11-24 shows the bit assignments of each register.

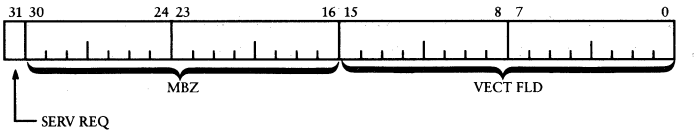


Figure 11-24 ▪ BR Receive Vector Register Format

Bit	Function		
31	SERV REQ (Service required) – Together with the vector value (bits 15-0), specifies the type of service that is required, as follows:		
	Bit 31	**Bits 15-0**	**Operation**
	0	0	No service required.
	0	V	UNIBUS service as indicated by vector (V) routine.
	1	0	UBA service as specified by the configuration and status register.
	1	V	UBA and UNIBUS service required. Save the vector, perform the service as specified by the configuration and status registers, and perform the routine as defined by the vector.
30:16	MBZ (Must be zero).		
15:0	VECT FLD (Vector field) – Specifies a vector field or a null vector (0).		

The UBA translates the UNIBUS interrupts to UBA interrupts, and the assertion of an SBI request line will initiate the interrupt transaction as specified by the vector address. The UBA checks to validate that the BR line corresponding to the BRRVR value is asserted, and that the BRRVR does not contain a previous vector. The UBA will then issue the UNIBUS bus grant and complete the UNIBUS interrupt transaction. The BRRVR is loaded with the interrupt vector at the successful completion of the interrupt transaction.

The device vector received during the transaction is transferred to the BRRVR by a read command. If a UBA interrupt is active, the vector is a negative quantity. The BRRVR is cleared by the successful completion of the SBI read data cycle or the vector is saved and the BRRVR remains full.

If the BRRVR value and the BR line do not correspond or the BRRVR contains a previous valid vector, then the contents of the BRRVR (either the stored vector from a previously failing SBI read data cycle or zero) are transferred by the read data operation. If a UBA interrupt is active, then the vector is sent as a negative value. The UNIBUS device interrupt sequence is performed in the following order.

1. A bus request line is asserted by the UNIBUS device.
2. The UBA asserts the SBI request line corresponding to the UNIBUS BR line, to initiate the interrupt transaction in the CPU.
3. When the interrupt summary read command that corresponds to the request level is detected by the UBA, the UBA asserts the request sublevel assigned to the UBA.
4. The CPU will transfer control to the UBA interrupt service routine.
5. The UNIBUS interrupt service routine executes a read to the BRRVR corresponding to the level of interrupt.
6. The UNIBUS adapter issues the UNIBUS bus grant (BG) corresponding to the level of the interrupt being serviced, providing the UBA interrupt is not pending; the BR line corresponding to the BRRVR is asserted; and the BRRVR does not contain a previous vector.
7. The UNIBUS interrupt transaction is then completed, the vector is loaded into the corresponding BRRVR and sent to the UNIBUS interrupt service routine by a read data operation, and the BRRVR is cleared when the confirmation is received.
8. The UNIBUS interrupt service routine will then transfer to the UNIBUS device service routine or service the UBA as indicated by the received interrupt vector.

If a device initiates but fails to complete an interrupt transaction, the interrupt vector will not be transferred into the interrupt vector register. When reading the interrupt vector register, the UNIBUS interrupt service routine receives a zero vector, which is the idle state of the BRRVR, and causes an error condition if desired. This terminates the service routine.

When the BRRVR is successfully loaded, it maintains the interrupt vector until a confirmation to the BRRVR read data sequence has been received or a UBA initialize sequence occurs. If the confirmation is not received, subsequent reads to the BRRVR result in the stored vector being returned for the data read.

When the UBA initiates an interrupt, it asserts the request line, resulting in an interrupt-summary read operation. The request sublevel assigned to the UBA is sent to the CPU as an interrupt summary response. Using the level and request sublevel, the CPU can dispatch the service routine, which reads the BRRVR corresponding to the level of interrupt. The BRRVR will contain

a negative value, and the service routine detects the value and branches to a routine that reads the configuration register and status register to determine the service required. The request line remains asserted until the bits of the UBA status register are cleared by the software.

▪ UBA REGISTER ASSIGNMENTS

The UBA registers occupy 16 pages (512 bytes per page) of SBI address space. The address location of the UBA is determined by the transfer request priority number assigned to the adapter. The transfer request number is selected by the position of jumper leads on the backplane. Table 11-8 lists the physical base address and SBI base address for a nexus assigned to any one of the SBI transfer request numbers. The VAX 8600 processor includes the SBIA 0 adapter; the SBIA 1 adapter is optional.

TABLE 11-8 ▪ Transfer Number Address Assignments

SBI Request TR level	Physical Base Address (hexadecimal) First SBI (SBIA 0)	Second (SBIA 1)
1	20002000	22002000
2	20004000	22004000
3	20006000	22006000
4	20008000	22008000
5	2000A000	2200A000
6	2000C000	2200C000
7	2000E000	2200E000
8	20010000	22001000
9	20012000	22012000
10	20014000	22014000
11	20016000	22026000
12	20018000	22018000
13	2001A000	2201A000
14	2001C000	2201C000
15	2001E000	2201E000

Table 11-9 lists each of the UBA registers and its associated physical address offset. The base address of the configuration register is the physical base address listed in table 11-8.

TABLE 11-9 • UBA Register Address Offset

UBA Register	Byte Offset (hexadecimal)
Configuration	000
Control	004
Status	008
Diagnostic Control	00C
Failed Map Entry	010
Failed UNIBUS Address	014
Failed Map Entry	018
Failed UNIBUS Address	01C
Buffer Selection Verification 0	020
Buffer Selection Verification 1	024
Buffer Selection Verification 2	028
Buffer Selection Verification 3	02C
Buffer Receive Vector 4	030
Buffer Receive Vector 5	034
Buffer Receive Vector 6	038
Buffer Receive Vector 7	03C
Data Path 0	040
Data Path 1-14	044-078
Data Path Register 15	07C
Reserved	080-7EC
Map 0	800
Map 1-494	804-EB8
Map 495	EBC
Reserved	EC0-EFC

- SBI ADDRESSABLE UBA REGISTERS

The UBA registers occupy eight pages of the SBI I/O address space. These registers are map registers, data path registers, interrupt vector registers, and control and status registers. The UBA registers are 32-bit registers and can be written only as longwords; however, they will respond to byte or word read commands. They also respond to the interlock-read/interlock-write sequence but will not affect the interlock of the SBI.

Configuration Register–The configuration register (CNFGR) contains the SBI fault bits, the UBA and UNIBUS environment status bits, and the UBA code. This register interfaces with the SBI. Figure 11-25 shows the format of the information in the register.

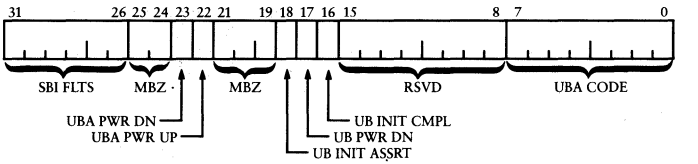


Figure 11-25 • Configuration Register Format

Bit	Function
31:26	SBI FLTS (SBI faults) – These bits are set when the UBA detects fault conditions on the SBI. The bits assignments are as follows. When bits 31:26 are set, the UNIBUS adapter asserts the fault signal on the SBI. Bit 31 (Parity fault) – Set when the UBA detects an SBI parity error. Bit 30 (Write-sequence fault) – Set when the UBA receives a write-masked, extended-write-masked, or an interlock-write-masked command that is not immediately followed by the expected data. Bit 29 (Unexpected read-data fault) – Set when the UBA receives data for which a read-masked, extended-read, or interlock-read-masked command has not been issued. Bit 28 (Interlock sequence fault) – Set when an interlock-write-masked command or a UNIBUS address space is received by the UBA without a previous interlock-read-masked command. Bit 27 (Multiple-transmitter fault) – Set when the IB bits transmitted by the UBA to the SBI are not the same as those received from the SBI. Bit 26 (Transmit fault) – Set if the UBA was the transmitter during a detected fault condition.
25:24	MBZ (Must be zero).
23	UBA PWR DOWN (Adapter powerdown) – Set when the ac voltage of the UBA power supply is below the specified value; cleared by writing a one to the bit location or when the adapter powerup (bit 22) is set.
22	UBA PWR UP (Adapter powerup) – Set when the ac voltage of the UBA power supply is normal; cleared by writing a one to the bit location or by setting the adapter powerdown (bit 23).
21:19	MBZ (Must be zero).

Bit	Function
18	UB INIT ASSRT (UNIBUS initialize asserted) – Set by the assertion of the UNIBUS INIT signal; cleared when the UNIBUS initialization complete (bit 16) is set or by writing a one to this bit location.
17	UB PWR DOWN (UNIBUS powerdown) – Set when UNIBUS AC LO signal is asserted and indicates that the UNIBUS has initiated a powerdown sequence. Cleared when the UNIBUS initialization complete (bit 16) is set or by writing a one to this location.
16	UB INIT CMPL (UNIBUS initialization complete) – Set by a successful completion of a powerup sequence on the UNIBUS.
15:8	RSVD (reserved).
7:0	UBA CODE – These bits define the code assigned to the UBA as follows: Bits 1 and 0 are determined by backplane jumpers and indicate the starting address of the UNIBUS address space associated with the UBA, and the value of 001010 is in bits 7 through 0.

Bits 0 1	Address Space (hexadecimal) UNIBUS	Starting
0 0	0	20100000
0 1	1	20140000
1 0	2	20180000
1 1	3	201C0000

UBA Control Register – The UNIBUS adapter control register (UACR) enables the software to control operations on both the UNIBUS adapter and the UNIBUS. All bits except the adapter initialize (bit 0) are set by writing a one and cleared by writing a zero to the bit location. Bit 0 is set by writing a one to the bit location. Figure 11-26 shows the bit format of the UACR.

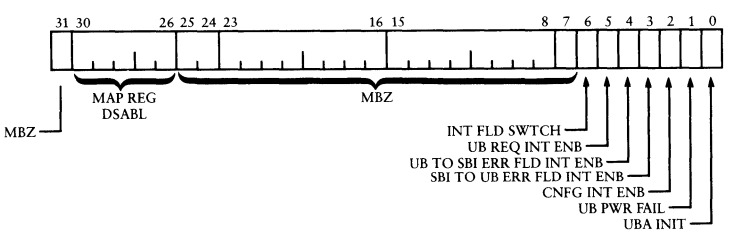


Figure 11-26 • Control Register Format

Bit	Function
31	MBZ (Must be zero).
30:26	MAP REG DSABL (Map register disable) – Five read/write bits that disable map registers in groups of 16 according to the binary value contained in the field. The disable bits prevent the double addressing of UNIBUS memory. DMA transfers to addresses controlled by disabled map registers are not recognized by the UBA, and no error bits are set and no transfers are initiated. The SBI, however, has access to disabled map registers. This field is initialized to zero and all map registers are enabled.

Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	UNIBUS Memory Kwords	Map Register Disabled
0	0	0	0	0	0	none
0	0	0	0	1	4	0-15
0	0	0	1	0	8	0-31
0	0	0	1	1	12	0-47
	(sequence continues to:)					
1	1	1	1	0	120	0-480
1	1	1	1	1	124	0-495

25:7	MBZ (Must be zero).
6	INT FLD SWTCH (Interrupt field switch) – Set to pass an interrupt from a device to the SBI, provided the BR interrupt enable (bit 5) is set. When cleared the interrupt is ignored.
5	UB REQ INT ENB (UNIBUS request interrupt enable) – Set by software to allow the UBA to pass interrupts from the UNIBUS to the CPU. This bit and bit 6 must be set to receive device interrupts.
4	UB TO SBI ERR FLD INT ENB (UNIBUS to SBI error field interrupt enable) – Set to enable an interrupt request to the CPU when any of the UNIBUS status register (UASR) bits (bits 10:9 or bits 7:3) are set during a DMA transfer.
3	SBI TO UB ERR FLD INT ENB (SBI to UNIBUS error field interrupt enable) – Set to allow the UBA to generate interrupt requests when one of the two UNIBUS timeout bits (0 or 1) is set in the UNIBUS status register (UASR).
2	CNFG INT ENB (Configuration interrupt enable) – Set at powerup and by program to allow the UBA to generate an interrupt request to the CPU when any one of the environmental status bits (bits 23:22 or bits 18:16) of the configuration register are set.

Bit	Function
1	UB PWR FAIL (UNIBUS powerfail) – Set to initiate a powerfail sequence on the UNIBUS; used by the software to initialize the UNIBUS. The UNIBUS remains powered down until this bit is cleared.
0	UBA INIT (UBA initialize) – Set to initialize the UBA and UNIBUS. The map registers, data path registers, status register, and control register will be cleared. The control logic will also be initialized and a powerfail sequence will occur on the UNIBUS. Only the configuration register and the diagnostic control register can be read from and only the configuration register, the diagnostic control register, and the control register can be written to during the adapter initialization sequence.

UBA Status Register–The UNIBUS adapter status register (UASR) is a read-only register that contains program status and error information used by the program during UNIBUS I/O operations. Figure 11-27 shows the register format. The register bits are defined as follows:

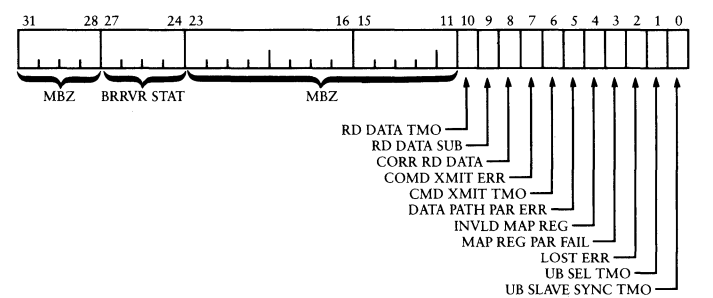


Figure 11-27 • UBA Status Register Format

Bit	Function
31:28	MBZ (Must be zero).
27:24	BRRVR STAT (Buffer receive vector register status) – Indicates the status of the four SBI addressable buffer receive vector registers (BRRVR). Each bit is set when the interrupt vector is loaded into the corresponding BRRVR during a UNIBUS interrupt transaction, providing that the SBI processor is acknowledging device interrupts. Cleared by the successful completion of a read data transmission following a read BRRVR command. The bit positions are assigned as follows: Bit 27 = BRRVR 7 full Bit 26 = BRRVR 6 full Bit 25 = BRRVR 5 full Bit 24 = BRRVR 4 full
23:11	MBZ (Must be zero).
10	READ DATA TMO (Read date timeout) – Set by the UBA when a device has initiated a DMA read transfer, after the following events have occurred: the UBA has successfully transmitted a read command to the SBI, the SBI memory has not returned the requested data within the specified interval, and the device has not timed out.
9	READ DATE SUB (Read data substitute) – Set if a read data substitute is received in response to a UNIBUS-to-SBI DMA read transfer. No data will be sent to the device, a device timeout occurs, and the nonexistent memory bit will be set in the device.
8	CORR READ DATA (Corrected read data) – Set by the UBA when it receives corrected read data in response to an SBI read command during a DMA read transfer.
7	COMD XMIT ERR (Command transmit error) – Set by the UBA when it receives an error confirmation in response to an SBI command transmission during a DMA transfer.
6	COMD XMIT TMO (Command transmit timeout) – Set when no response is received from a command for a UNIBUS-to-SBI data transfer, for a buffered-data-path-to-SBI write operation, or for a purge operation.
5	DATA PATH PAR ERR (Data path parity error) – Set when a parity error occurs in the buffered data path during either a DATI transfer from the UNIBUS to the buffered data path, a buffered-data-path-to-SBI write operation or during the purge operation.

Bit	Function
4	INVLD MAP REG (Invalid map register) – Set by the UBA during a DMA transfer or clear-and-initialize operation when the UNIBUS address points to a map register that has not been validated by the software or when the DMA transfer crosses an SBI page boundary for which the map register has not been validated.
3	MAP REG PAR FAIL (Map register parity failure) – Set when a UNIBUS address is being mapped to an SBI address on a DMA transfer operation or during a purge operation.
2	LOST ERR (Lost error) – Set by the UBA if the locking error field is locked and another error within this field occurs. The lost error bit does not initiate an interrupt request.
1	UB SEL TMO (UNIBUS select timeout) – Set by the UBA if it cannot gain access to the UNIBUS within 50 microseconds during the execution of a software-initiated SBI-to-UNIBUS transfer. This condition indicates the presence of a hardware failure on the UNIBUS.
0	UB SLAVE SYN TMO (UNIBUS slave synch timeout) – Set when a response is not received for SBI-to-UNIBUS software-initiated transfer during the data transfer cycle on the UNIBUS. This condition indicates a transfer failure when a nonexistent memory or device on the UNIBUS is addressed.

Diagnostic Control Register – The diagnostic control register (DCR) is a read-write register that provides control and status information to aid in testing and diagnosing the UBA during maintenance functions. Figure 11-28 shows the bit format of the DCR.

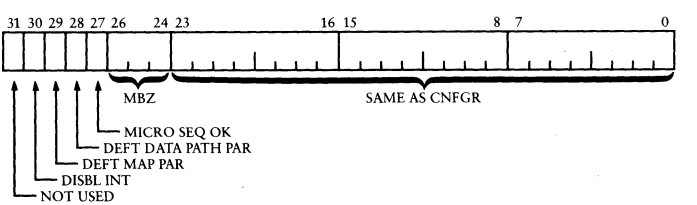


Figure 11-28 • Diagnostic Control Register Format

Bit	Function
31	Not used.
30	DISBL INT (Disable interrupt) – Set to prevent the UBA from recognizing interrupts on the UNIBUS. Used for testing the response of the UBA to the passive release condition during a UNIBUS interrupt transaction. Set by writing a one and cleared by writing a zero to the bit location.
29	DEFEAT MAP PAR (Defeat map parity) – A read-write bit set to prevent the parity bits of the map registers from entering the map-register parity checkers. The map register parity generator checkers generate and check parity on 8-bit quantities. Each parity field is 8 data bits and 1 parity bit and the total number of ones in the parity field is always an odd number.
28	DEFEAT DATA PATH PAR (Defeat data path parity) – Set to inhibit the parity bits of the data path RAM from entering the parity checkers. The data path parity generator checkers generate and check parity on 8-bit data units. The total number of ones in the parity field is always an odd number. When the integrity of the parity generator checkers is to be tested, the total number of ones in at least one of the bytes of data must be even. With the parity bit disabled, a data path parity failure will result during a DMA transfer via that buffered data path.
27	MICRO SEQ OK (Microsequencer OK) – A read-only bit that indicates that the UBA microsequencer is in the idle state after it has completed the initialization sequence or once it has completed a UBA function. This bit can be used by diagnostics to determine whether the microsequencer has completed a successful powerup sequence or remains in a loop.
26:24	MBZ (Must be zero).
23:0	SAME AS CNFGR – Same function as bits 23:0 of the configuration register shown in figure 11-25.

Failed Map Entry Register – The failed map entry register (FMER) is a read-only register that contains the map register number used for either a DMA transfer or a purge operation that has resulted in the setting of one of the following error bits of the status register: INVLD MAP REG, MAP REG PAR, DATA PATH PAR ERR, COMM XMIT ERR, COMM XMIT ERR, RD DATA SUB, and RD DATA TMO. This register is locked and unlocked by the UNIBUS-to-

SBI data transfer error field of the UASR. The contents are valid only when the register is loaded. The software can read this register to obtain the map register number associated with the failure and to read the contents of the map register to determine the number of the data path that failed. Figure 11-29 shows the format of the information in the register.

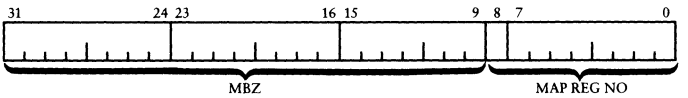


Figure 11-29 • Failed-Map-Entry Register Format

Bit	Function
31:9	MBZ (Must be zero).
8:0	MAP REG NO (Map register number)—These bits contain the number of the map register in use at the time of a failure. They correspond to bits 17:9 of the UNIBUS address.

Failed UNIBUS Address Register—The failed UNIBUS address register (FUBAR) is a read-only register that contains the upper 16 bits of the UNIBUS address translated from an SBI address during a previous software-initiated data transfer. The UB SEL TMO and UB SLAVE SYNC TMO errors of the UNIBUS status register will lock this register. When the error bit is cleared, the register will be unlocked. Figure 11-30 shows the bit format of the register.

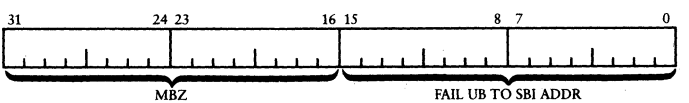


Figure 11-30 • Failed-UNIBUS-Address Register Format

Bit	Function
31:16	MBZ (Must be zero).
15:0	FAIL UB TO SBI ADDR (Failed UNIBUS to SBI address) – These bits correspond to UNIBUS address bits A17:A2

Buffer Selection Verification Registers – The buffer selection verification registers (BRSVR) are four read-write registers that provide the diagnostic software access for testing the integrity of the data path RAM. Four locations in the data path RAM have been assigned to these registers. Writing and reading the BRSVR has no effect on the UBA. The BRSVR bit configuration is shown in figure 11-31.

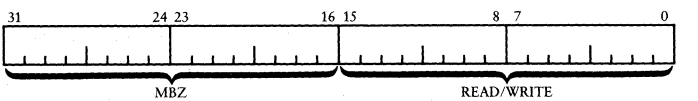


Figure 11-31 • Buffer-Selection-Verification Register Format

Bit	Function
31:16	STZ (Set to zero).
15:0	TEST DATA – Read/write test data.

BR Receive Vector Registers – The BR receive vector registers (BRRVR) are four read-only registers, BRRVR 7 through BRRVR 4. They correspond to UNIBUS-interrupt bus request levels 7 through 4 respectively. Each BRRVR is a read-only register and contains the interrupt vector of a UNIBUS device interrupting at the corresponding BR level. The BRRVR is read by the software during the UBA interrupt service routine. The contents of the BRRVR are used by the software to determine if a UBA interrupt is pending. Figure 11-32 shows the format of the register.

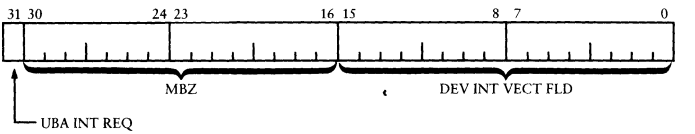


Figure 11-32 • BR Receive Vector Register Format

Bit	Function
31	UBA INT REQ (Adapter interrupt request indicator) – Set to indicate a UBA interrupt is pending.
30:16	MBZ (Must be zero).
15:0	DEV INT VECT FLD (Device interrupt vector field) – Contains the device interrupt vector loaded by the UBA during a UNIBUS interrupt transaction.

Data Path Registers – The UBA contains 16 data path registers (DPR0 through DPR15), which are used to transfer the information through the UBA. Each register corresponds to one of the 16 data paths. The DPRs are 32-bit read-write registers and the bit assignment of the register information is shown in figure 11-33.

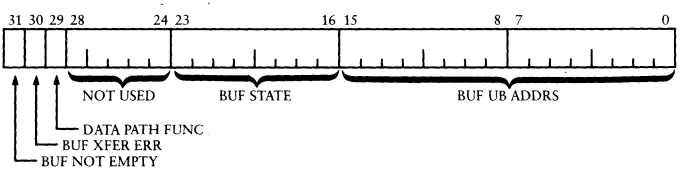


Figure 11-33 • Data Path Register Format

Bit	Function
31	BUF NOT EMPTY (Buffer not empty) – Set to indicate the buffer status, as follows: 0 = buffer is empty or does not contain valid data 1 = buffer contains valid data bit
30	BUF XFER ERR (Buffer transfer error) – Set by the UBA to indicate that a failure has occurred during a DMA write transfer or during a purge operation. Also set when a data path parity failure has occurred during a DMA read transfer. Any additional DMA transfers through this buffer will be aborted until the bit is cleared by the software. If a UNIBUS parity error occurs during a DMA read transfer, the UNIBUS PB signal line will be asserted to enable the device to abort the transfer. The purge operation does not clear this bit.
29	DATA PATH FUNC (Data path function) – A read-only bit that indicates the function of the DMA transfer as follows: 0 = DMA read 1 = DMA write
28:24	Not used.
23:16	BUF STATE (Buffer state) – Used by the diagnostic program and indicate the state of each of the eight buffers of the associated data path during a DMA write transfer. The UBA generates the SBI mask bits from these bits during a DMA write transfer or purge operation. The bits are set as each byte is written from the UNIBUS and cleared during the SBI write operation. They specify the following: 0 = buffer empty 1 = buffer full
15:0	BUF UB ADDR (Buffered UNIBUS address) – If the transfer is in the byte offset mode and the last UNIBUS transfer has entered into the next quadword, these bits contain the upper 16 bits of the UNIBUS address (A17:A2) asserted during the DMA transfer through the data path. This is the UNIBUS address from which the SBI address will be mapped if a purge operation occurs before the next UNIBUS transfer.

Map Registers—The UBA contains 496 read-write map registers (MR0 through MR495), one for each UNIBUS memory page address. The register assignments for each offset are as follows:

Register	Offset
MR0	800
MR1	804
MR2	808
MR3	80C
(sequence continues to:)	
MR494	FB8
MR495	FBC

The upper nine address bits (A17:A9), asserted by the device during a DMA transfer, select the map register that was set up by the software. The map register tests the validity of the current UNIBUS transfer, selects one of the 16 data paths for the transfer, determines if the transfer will occur in the byte-offset mode, and maps the UNIBUS page address to an SBI page address. Figure 11-34 shows the map register bit configuration.

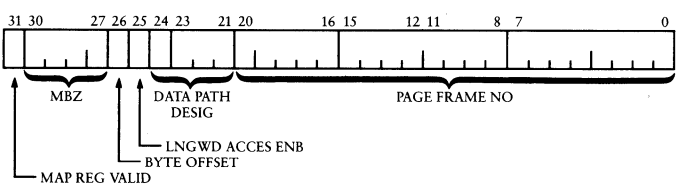


Figure 11-34 • Map Register Format

Bit	Function
31	MAP REG VALID (Map register valid)—Controlled by the software to indicate the status of the map register contents. This bit is monitored each time the UNIBUS memory page is accessed. When set, the UNIBUS transfer is allowed to continue. When cleared, the UNIBUS transfer is aborted, a nonexistent memory error is indicated to the device, and the invalid map register bit is set in the UBA status register.

Bit	Function
30:27	MBZ (Must be zero).
26	LNGWD ACCES ENB (Longword access enable)—When set, and the map register selects a buffered data path, then the longword-aligned 32-bit random access mode is enabled. This bit is cleared during the UBA initialization.
25	BYTE OFFSET—When set, and when the UNIBUS memory page corresponding to this register is using one of the buffered data paths and the transfer is to an SBI memory address, then the UNIBUS adapter will perform a byte offset operation on the current UNIBUS data transfer. In the VAX 8600 the condition of a transfer to the SBI memory address is not required. The software can interpret this operation as an increase in the physical SBI memory address, mapped from the UNIBUS address, by one byte. This allows word-aligned UNIBUS devices to transfer to odd-byte memory addresses. This bit is cleared during the UBA initialization.
24:21	DATA PATH DESIG (Data path designator)—A binary number, selected by software that indicates the data path to be used by the UNIBUS memory page. More than one UNIBUS transfer can be assigned to the direct data path and only one active transfer is assigned to a buffered data path. The designator bits are cleared during UBA initialization and the values are assigned as follows:
	Value **Data Path**
	0 Direct data path 1-15 Buffered data paths 1 through 15
20:0	SBI PAGE ADDR (SBI page address)—The SBI page address (or page frame number) to which the UNIBUS memory page will be mapped. These bits perform the UNIBUS-to-SBI page address translation. When an SBI transfer is initiated, the contents of SBI page address (SPA 27:7) are concatenated with UNIBUS address bits (bits A8:A2) to form the 28-bit SBI address.

MASSBUS Subsystem Functions

The MASSBUS subsystem is a dedicated communications path that allows high-performance mass storage devices including disk drives and magnetic tape transports to be interfaced with medium- and large-VAX systems. The subsystem includes an RH750 MASSBUS adapter (MBA) for the VAX-11/750 processor and an RH780 MBA for the VAX-11/780 series processors

and the VAX 8600 processor. Included with the MBA is the MASSBUS, which forms the data and control path for the devices. The MBA is the hardware link between the internal processor bus and the MASSBUS lines. In the VAX-11/750 system, the adapter connects to the computer memory interconnect (CMI) bus within the processors; in large VAX processors the adapter connects to the synchronous backplane interconnect (SBI) bus of the processor. The MASSBUS adapters perform similar functions with both medium- and large-VAX processors; however, the methods used to implement these functions can vary, depending on the adapter.

• MASSBUS SIGNAL LINES

Figure 11-35 shows the signal lines that connect the devices with the MBA. Table 11-10 lists each of the lines and defines their function. The MASSBUS consists of 54 signal lines, including asynchronous control path and synchronous data path lines.

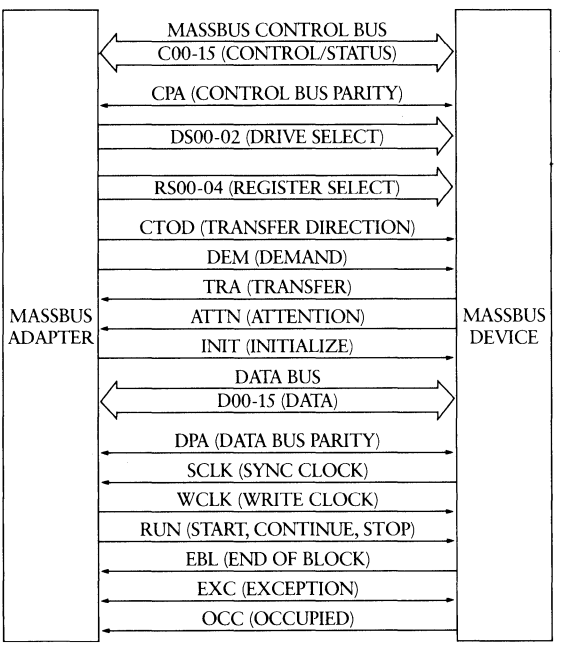


Figure 11-35 • MASSBUS Signal Lines

TABLE 11-10 • MASSBUS Line Descriptions

Line	Description
C00-C15	Control and status – Transfers 16 parallel control or status bits to or from the device.
CPA	Control bus parity – Transfers odd control bus parity to or from the device. The parity is simultaneously transferred with control bus data.
DS00-DS02	Device select – Transfers a 3-bit binary code from the MBA to a select controller. The device responds when the unit select switch in the controller corresponds to the transmitted binary code.
RS00-RS04	Register select – Transfers a 5-bit binary code from the MBA to select a particular device register.
CTOD	Controller-to-device – Indicates the direction of information transfer on the control bus. For a controller-to-device transfer, the MBA asserts CTOD, and for a device-to-controller transfer, the MBA negates CTOD.
DEM	Demand – Asserted by the MBA to indicate that a transfer will occur on the control bus. For a controller-to-drive transfer, the line is asserted by the MBA when data is present. For a drive-to-controller transfer, the line is asserted by the MBA to request data and is negated when the data has been received from the control bus. The RS, DS, and CTOD lines are asserted before the assertion of this line.
TRA	Transfer – Asserted by the device in response to the assertion of the DEM line. For a controller-to-drive transfer, the TRA line is asserted when the data is strobed, and negated when the DEM line is negated. For a drive-to-controller transfer, TRA line is asserted when the data is asserted on the bus and negated when the DEM line is negated.
ATTN	Attention – This line is connected to all devices on the MASSBUS and asserted by the device to inform the MBA of any change in device status or of an abnormal condition. This line is asserted when the ATA status bit of a device is set and may be asserted by more than one device at a time.
INIT	Initialize – Asserted by the MBA to initialize all devices on the bus. This signal is transmitted whenever the MBA receives an initialize command.

Line	Description
FAIL	Fail – Asserted to indicate that a powerfail condition has occurred in the MBA or that the MBA is in maintenance mode.
D00-D15	Data – Bidirectional lines that transfer 16 bits of parallel data between the MBA and devices.
DPA	Data bus parity – Transfers an odd parity bit to or from the device simultaneously with the data on the data lines.
SCLK	Sync clock – Asserted by the device during a read operation to indicate that the data is to be strobed by the MBA. During a write operation, SCLK is asserted to the MBA to indicate the rate at which data on the data lines would be presented by the MBA on the data bus.
WCLK	Write clock – Asserted by the MBA to indicate that the data on the data lines is available for the device.
RUN	Run – Asserted by the MBA to initiate the data transfer command execution. During a data transfer, the device samples the line at the end of each sector and, if it is asserted, the drive continues the transfer into the next sector. If RUN is negated, the device terminates the transfer.
EBL	End-of-block – Asserted by the device at the end of each sector. For certain error conditions when the operations must be terminated immediately, this line is asserted prior to the normal time and the transfer is terminated before the end of the sector.
EXC	Exception – Asserted by the device or MBA to indicate that an error condition has occurred during a data transfer command. This line remains asserted until the trailing edge of the last end-of-block signal occurs.
OCC	Occupied – Indicates acceptance of a valid data transfer command.

- MASSBUS ADAPTER OPERATION

The MASSBUS adapter (MBA) consists of an SBI/MBA interface, internal registers, control paths, and data paths. Figure 11-36 is a simplified diagram of the MBA. An internal bus connects the SBI module to the internal registers, control paths, and data paths, and provides for the passage of data to the various functional blocks.

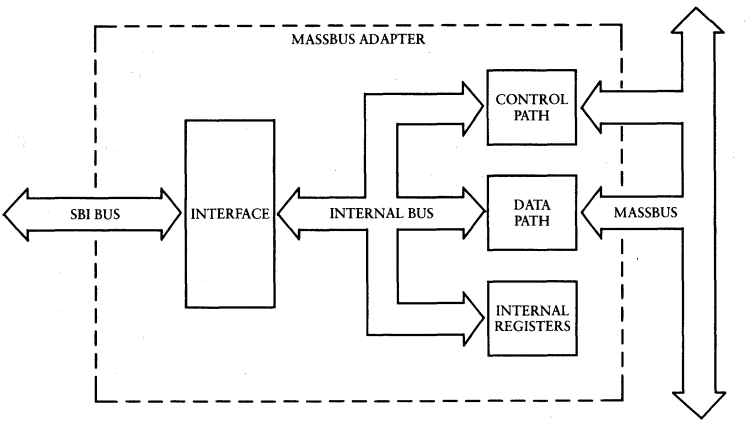


Figure 11-36 ▪ MASSBUS Adapter Configuration

The MBA accepts and executes commands from the CPU and reports status changes and fault conditions to the CPU. The MBA transfers register data or blocks of data to or from mass storage devices. A 256 by 32-bit RAM in the MBA is a map register that stores the physical page addresses of the block of data to be transferred. Information can be transferred to or from contiguous or noncontiguous memory locations. Memory data is transferred through the SBI to the MBA as a 64-bit quadword and between the MBA and a device as eight 16-bit words. The MBA can transfer data to or from a device at a rate of 16 bits per microsecond. The MBA controls the transfer of data between the device and the SBI and contains a 32-byte buffer used to store the data for transfer. Four MASSBUS transfers of 16 bits each are therefore required for each memory transfer. The MBA accepts longword-aligned read and write transactions to the internal registers.

The control path is used for the transfer of control information to and from the MASSBUS devices. Sections of the MBA address space select registers physically located within MASSBUS devices. The MASSBUS control path is used to communicate with these data path registers.

The data path controls the data transferred to and from the MASSBUS device and the SBI. The 32-bit SBI data word is divided into 16-bit segments. When performing a read operation from a MASSBUS device, the data path assembles the two 8-bit bytes from the MASSBUS into the 32-bit SBI format. A silo and input/output data buffer are used to increase the data transfer rate. The data path also contains a write check circuit, which can be used under program control to verify the accuracy of the data transfer function.

Each SBI device is a nexus and is assigned an 8-Kbyte control address space that is accessible as part of the SBI I/O longword address space. The command/address formats used to access the medium- and large-VAX system MBA registers is shown in figure 11-37.

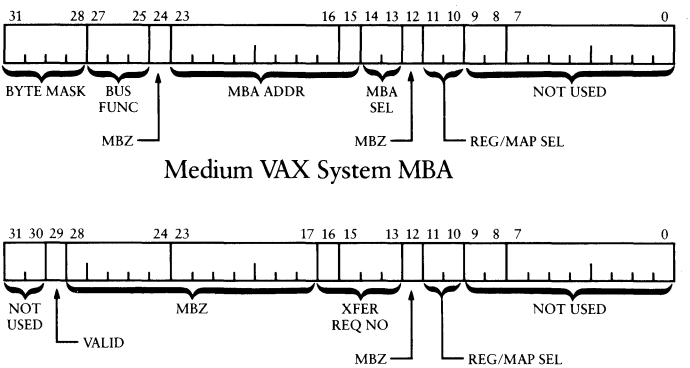


Figure 11-37 • Command/Address Word Format

Medium VAX System MBA	
Bit	**Function**
31:28	BYTE MASK.
27:25	BUS FUNC (Bus function).
24	MBZ (Must be zero).
23:15	MBA ADDR (MBA address) – A hexadecimal value (1E15) used to select the MBA address space.
14:13	MBA SEL (MBA Select) – A binary value that selects the MBA addressed.
12	MBZ (Must be zero).
11:10	REG/MAP SEL (Register or map select) – A binary value that selects one of the following I/O address spaces:

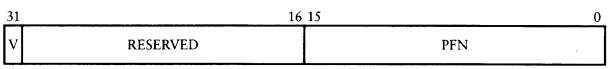
Bit	Function		
	Bit 11	**Bit 10**	**Selection**
	0	0	MBA internal register Bits 9:5 must be zero Bits 4:2 register select offset
	0	1	MBA external register Bits 9:7 device select Bits 6:2 register select
	1	0	MBA map Bits 9:2 map address
	1	1	Invalid – No response to address
9:0	Not used.		

Large VAX System MBA

Bit	Function		
31:30	Not used.		
29	VALID – Must be set to 1.		
28:17	MBZ (Must be zero).		
16:13	XFER REQ NO (Transfer request number).		
12	MBZ (Must be zero).		
11:10	REG/MAP SEL (Register or map select) – A binary value that selects one of the following I/O address spaces:		
	Bit 11	**Bit 10**	**Selection**
	0	0	MBA internal register Bits 9:5 must be zero Bits 4:0 register select offset
	0	1	MBA external register Bits 9:7 device select Bits 6:0 register select
	1	0	MBA map Bits 9:0 map address
	1	1	Invalid – No response to address
9:0	Not used.		

• VIRTUAL TO PHYSICAL ADDRESS TRANSLATION

The virtual address register (VAR) and the 256 map registers are used to transform the virtual addresses from memory into the physical addresses of the registers and storage locations of the MBA and devices on the MASSBUS. The virtual address is loaded into the VAR before the data transfer is initiated. The register stores the address information from the SBI and uses it to access the data in the map register file. The address translations performed between medium and small systems are similar. Figure 11-38 shows the translation process for medium-VAX processors and figure 11-39 shows the process for large-VAX processors.



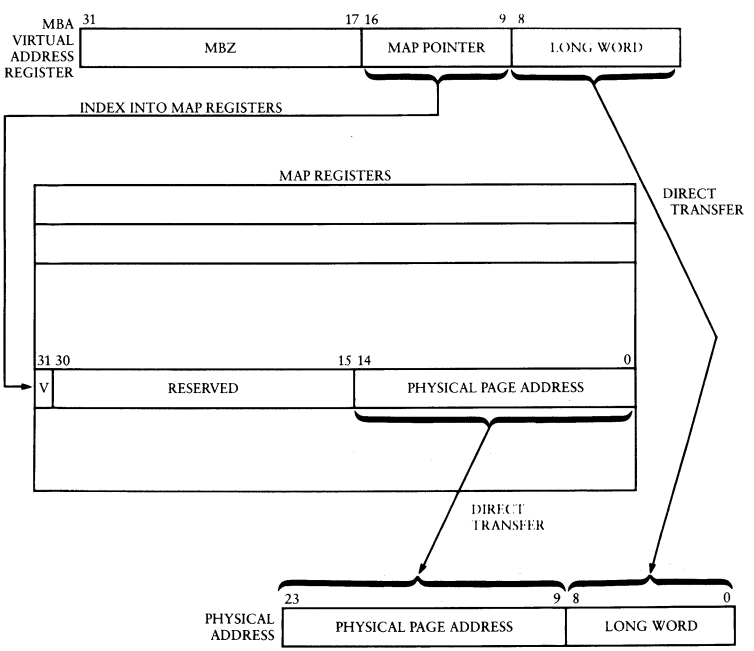


Figure 11-38 • Virtual-to-Physical Address Translation (Medium-VAX Processor)

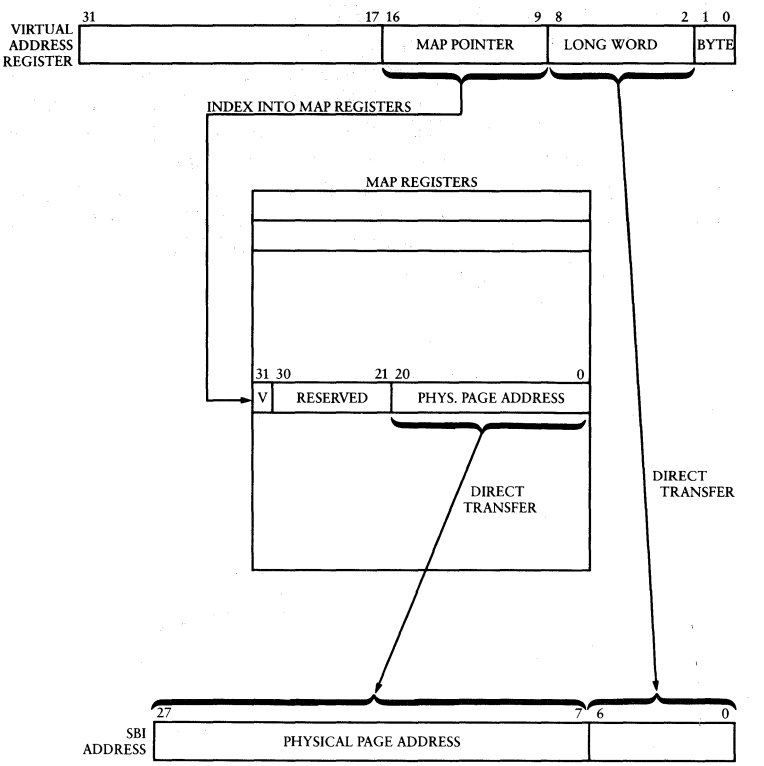


Figure 11-39 ▪ Virtual-to-Physical Address Translation (Large-VAX Processors)

▪ MBA REGISTERS

The MBA address space contains internal and external registers. The internal registers are located in the MBA, and the external registers are drive and drive-controller registers located in the MASSBUS devices. The internal registers, including the 256 by 32-bit map registers, monitor the status conditions and control the adapter operation and data transfers between the SBI and the devices. These functions include maintaining a byte count to ensure that all of the data to be transferred has been accounted for, and converting virtual addresses to physical addresses for referencing data in memory. The number of internal registers depends on the type of MBA. Table 11-11 lists the internal MBA registers.

TABLE 11-11 • MBA Internal Registers

Register Name
Control register (CR)
Status register (SR)
Byte count register (BCR)
Diagnostic register (DR)
Command/address register (CAR)
Virtual address register (VAR)
Configuration register (CSR)*
Selected map register (SMR)*
*Contained in MBA for large-VAX processors only

Virtual Address Register—The map-select and byte-offset values of the virtual address register (VAR) are used to assemble a physical SBI address to be sent to memory. The VAR is incremented by eight after every memory read or write operation. Figure 11-40 shows the format of the register information.

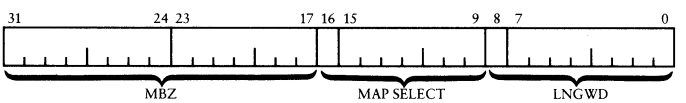


Figure 11-40 • Virtual Address Register Format

FIGURE 11-40 • Virtual Address Register Format

Bit	Function
31:17	MBZ (Must be zero).
16:9	MAP SELECT—A binary value used to select one of 256 MBA map registers.
8:0	LNGWD—A binary value used to select the byte offset into the page of the current data byte.

Control Register–The control register (CR) is a read-write register with a byte offset value of four and is used to control the operations of the MBA. Figure 11-41 shows the CR bit configurations. The bit functions are described as follows:

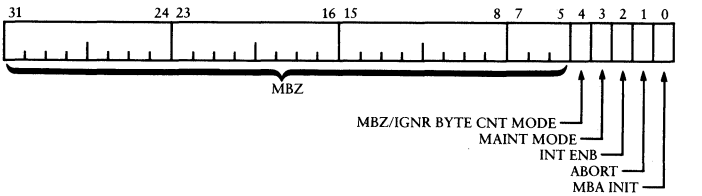


Figure 11-41 ▪ Control Register Format

Bit	Function
31:5	MBZ (Must be zero).
4	MBZ (Must be zero for large-VAX processors).
4	IGNR BYTE CNT MODE (Ignore byte count mode)–For the VAX-11/750 processors, this bit is set to prevent the termination of a data transfer as a result of a byte counter overflow. The data transfer is terminated when a signal informs the MBA that the last byte has been transferred. It is cleared by writing a zero to this bit or by MBA initialize.
3	MAINT MODE (Maintenance mode)–Set to select the maintenance mode for the MBA operation. This mode allows the diagnostic program to exercise and examine the MASSBUS without the use of MASSBUS devices. The current MASSBUS operation must be completed before this mode can be initiated.
2	INT ENB (Interrupt enable)–Set to allow the MBA to interrupt the CPU when specific conditions occur.
1	ABORT–Set to stop the data transfer operations and will cause an interrupt to the processor if the INT ENB (bit 2) is set.
0	MBA INIT (MBA initialize)–A write-only bit set to initialize the MBA by clearing specific registers within the MBA, by canceling the commands pending, and by aborting data transfers.

Status Register–The status register (SR) is a read-write register that contains the status of the MBA and can be monitored by the processor or devices. The functions of the SR for medium- and large-VAX processors are similar. The differences are listed in the bit descriptions. Figure 11-42 shows the assignments of the bits in the register.

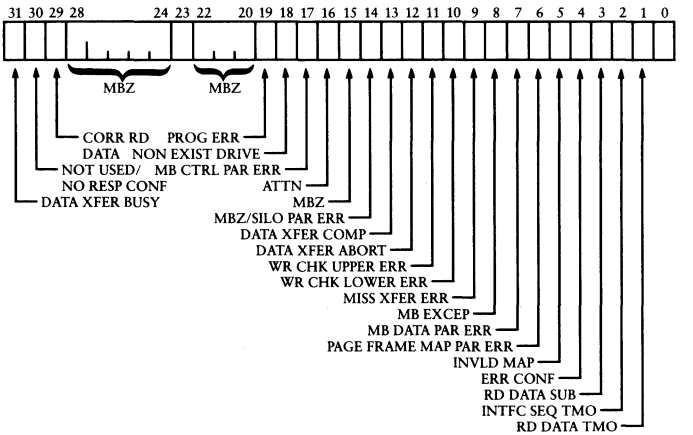


Figure 11-42 • Status Register Format

Bit	Function
31	DATA XFER BUSY (Data transfer busy)–Set when a data transfer command is received and cleared when the transfer has been completed.
30	Not used with VAX-11/750 processor.
30	NO RESP CONF (No response confirmation)–In large VAX processors, this bit is set when no response confirmation is received by the MBA for a read or write command or when write data is sent to the SBI. Setting this bit will cause the command to be reissued. The bit is cleared by writing a one and by the MBA initialize sequence.
29	CORR READ DATA (Corrected read data)–Set when the data received by the MBA from memory has been corrected; cleared by the subsequent receipt of a valid data transfer command, by writing a one to this bit, or by the MBA initialize sequence.
28:20	MBZ (Must be zero)–For large-VAX processors.

Bit	Function
28:24	MBZ (Must be zero) – For VAX-11/750 processor.
23	CNTL BUS HUNG (Control bus hung) – Set on the VAX-11/750 processor to indicate that the transfer line (TRA) has been asserted for 1.5 microseconds after the MBA has received a read or write command to an external register, and that any previous external register operations are complete. No data transfer operations can be initiated until this bit is cleared by writing a one to this bit or by the MBA initialize sequence.
22:20	MBZ (Must be zero).
19	PROG ERR (Programming error) – Set when the MBA is performing a data transfer and the program attempts to initiate a transfer; to load the map register, VAR, or BAR; or to select the maintenance mode. Setting this bit will initiate a program interrupt if the INT ENB bit of the CR is set. It is cleared by writing a one to this bit or by the MBA initialization sequence.
18	NON EXIST DRIVE (Nonexistent drive) – Set when a device fails to assert the transfer (TRA) line within 1.5 microseconds after the assertion of the demand (DEM) line. Setting this bit will cause all zeros to be sent to the SBI and will interrupt the processor if the INT ENB (bit 2) of the CR is set. Cleared by writing a one to this bit or by the MBA initialization sequence.
17	MB CNTL PAR ERR (MASSBUS control parity error) – Set when a MASSBUS control parity error occurs; cleared by writing a one to this bit or by the MBA initialization sequence.
16	ATTN (Attention) – Set when the attention line is asserted; causes a program interrupt if the INT ENB (bit 2) of the CR is set.
15:14	MBZ (Must be zero) – For large-VAX processors.
15	MBZ (Must be zero) – For the VAX-11/750 processor.
14	SILO PAR ERR (Silo parity error) – Set on the VAX-11/750 processor when a silo parity error occurs during a data transfer; stopping the transfer. This results in a processor interrupt if the INT ENB (bit 2) of the CR is set. It is cleared when the MBA subsequently receives a valid data command, by writing a one to the bit, or by the MBA initialize sequence.
13	DATA XFER COMP (Data transfer complete) – Set when the data transfer is completed normally or because of an error. It is cleared when the subsequent valid data command is received, by writing a one to this bit, or by the MBA initialization sequence.

Bit	Function
12	DATA XFER ABORT (Data transfer aborted) – Set by the signal on the end-of-block (EBL) line when the data transfer has been aborted. An interrupt request is initiated if the INT ENB (bit 2) of the CR is set. It is cleared by writing a one to this bit or by the MBA initialization sequence.
11	DATA LATE – Set by the write-clock signal when the data buffer is empty during a write-data or write-check-data transfer, or by the sync-clock signal when the data buffer is full during a read data transfer. This will abort the data transfer. It is cleared by writing a one to this bit or by the MBA initialization sequence.
10	WR CHK UPPER ERR (Write check upper error) – Set when a compare error is detected in the upper byte of data while the MBA is performing a write check operation; cleared by writing a one to this bit, or by the MBA initialization sequence.
9	WR CHK LOWER ERR (Write check lower error) – Set when a compare error is detected in the lower byte of data while the MBA is performing a write check operation; cleared by writing a one to this bit or by the MBA initialization sequence.
8	MISS XFER ERR (Miss transfer error) – Set when the sync-clock or occupied signal is not received within 500 microseconds after data transfer busy (bit 31) is set. This initiates an interrupt request if the INT ENB (bit 2) of the CR is set. It is cleared by writing a one to this bit or by the MBA initialization sequence.
7	MB EXCEP (MASSBUS exception) – Set when the exception signal is received from the MASSBUS on the EXC line and results in the termination of the data transfer.
6	MB DATA PAR ERR (MASSBUS data parity error) – Set when a MASSBUS data parity error is detected during a read data transfer operation and results in the termination of the data transfer; cleared by writing a one to this bit or by the MBA initialization sequence.
5	PAGE FRAME MAP PAR ERR (Page frame map parity error) – Set when a parity error is detected on the page frame number that was read from the page frame map. This results in the termination of the data transfer. It is cleared by writing a one to this bit or by the MBA initialization sequence.
4	INVLD MAP (Invalid map) – Set when the valid bit of the next page frame number is zero, and will terminate the data transfer operation; cleared by writing a one to this bit or by the MBA initialization sequence.

Bit	Function
3	ERR CONF (Error confirmation) – Set when the MBA receives an error confirmation for a read or write command, resulting in the termination of the data transfer operation; cleared by writing a one to this bit or by the MBA initialization sequence.
2	RD DATA SUB (Read data substitute) – Set when the tag of the data read from memory indicates a read data substitute, resulting in the termination of the data transfer operation; cleared by writing a one to this bit or by the MBA initialization sequence.
1	INTFC SEQ TMO (Interface sequence timeout) – Set when no response is received within 102.4 microseconds after the SBI sequence has begun and one of the following conditions has occurred. This results in the termination of the data transfer operation. – An acknowledge is received for a command or address transfer that specifies a read operation. – An acknowledge is received for a command address transfer that specifies a write operation; an acknowledge is also received for each transmission of write data. – An error confirmation signal is received for a command or address transfer.
0	RD DATA TMO (Read data timeout) – Set when the time between the initiation of the read command and the reception of the data by the requesting nexus has exceeded 102.4 microseconds.

Byte Count Register – The byte count register (BCR) is a read-write register that contains the number of bytes to be transferred during a read or write data sequence. Figure 11-43 shows the location of the byte count information of the BCR.

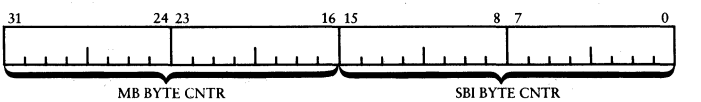


Figure 11-43 ▪ Byte Count Register Format

Bit	Function
31:16	MB BYTE CNTR (MASSBUS byte counter) – Read-only bits loaded by the MBA serve as the counter for the number of bytes transferred through the SBI interface.
15:0	SBI BYTE CNTR (SBI byte counter) – Read-write bits that contain the two's complement of the number of bytes to be transferred during the read or write operation.

Diagnostic Register – The diagnostic register (DR) is a read-write register used during the maintenance mode of operation. The register allows error conditions to be programmed into the MBA and the results to be evaluated to determine the location of failures. Figure 11-44 shows the configuration of the bits in the diagnostic register.

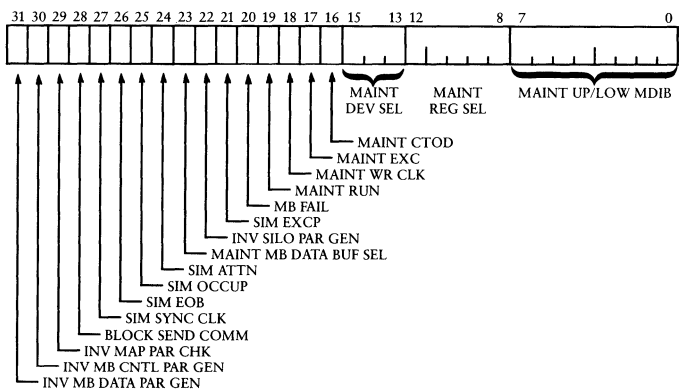


Figure • 11-44

Bit	Function
31	INV MB DATA PAR GEN (Invert MASSBUS data parity generator) – Set to invert the data parity generator bit.
30	INV MB CNTL PAR GEN (Invert MASSBUS control parity generator) – Set to invert the control parity generator bit.
29	INV MAP PAR CHK (Invert map parity checking) – Set to invert the map parity check bit.

Bit	Function
28	BLK SEND COMM (Block sending command) – Set during a data transfer to cause a data late (bit 11) of the SR to be set and a program interrupt to occur.
27	SIM SYNC CLK (Simulate sync clock) – Writing a one to this bit simulates the assertion and writing a zero simulates the negation of the synchronize clock line when the maintenance mode (bit 3) of the CR is set.
26	SIM EOB (Simulate end of block) – Writing a one simulates the assertion and writing a zero simulates the negation of the end-of-block line when the maintenance mode (bit 3) of the CR is set.
25	SIM OCC (Simulate occupied) – Writing a one to this bit simulates the assertion and writing a zero simulates the negation of the occupied line when the maintenance mode (bit 3) of the CR is set.
24	SIM ATTN (Simulate attention) – Writing a one to this bit simulates the assertion and writing a zero simulates the negation of the attention line when the maintenance mode (bit 3) of the CR is set.
23	MAINT MB DATA BUF SEL (Maintenance MASSBUS data buffer select) – Determines the information to be sent from bits 15:8 of this register. It is set to select the upper byte (bits 15:8) of this register and is cleared to select the maintenance drive and register select information.
22	INV SILO PAR GEN (Invert silo parity generator) – Invert the silo parity generator bit.
21	SIM EXC (Simulate exception) – Writing a one to this bit simulates the assertion and writing a zero simulates the negation of the exception line when the maintenance mode (bit 3) of the CR is set.
20	MB FAIL (MASSBUS fail) – Set when the maintenance mode (bit 3) of the CR is set.
19	MAINT RUN (Maintenance run).
18	MAINT WR CLK (Maintenance write clock).
17	MAINT EXC (Maintenance exception).
16	MAINT CTOD (Maintenance controller to drive).
15:13	MAINT DEV SEL (Maintenance device select).
12:8	MAINT REG SEL (Maintenance register select).
7:0	MAINT UP/LOW MDIB (Maintenance upper/lower information bus).

Diagnostic Register Format

Selected Map Register–The selected map register is a read-only register that has the same format as the map register and is valid only when the data transfer busy (bit 31) of the status register is set.

- CONFIGURATION/STATUS REGISTER

The configuration/status register (CSR) is a read-write register that provides status and configuration information of the MBA installed in large VAX systems. Figure 11-45 shows the format of the information in the register.

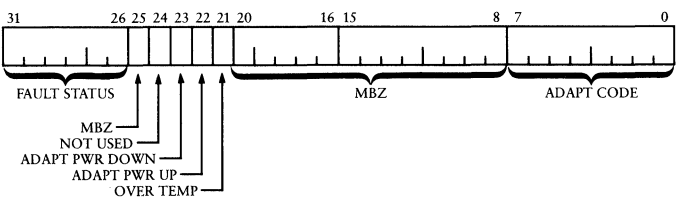


Figure 11-45 • Configuration/Status Register Format

Bit	Function
31:26	FLT STATUS (Fault status)–Indicates the following fault status conditions that result in the assertion of a fault line of the SBI.
	Bit 31 (SBI parity error)–Set when an SBI parity error has been detected. Cleared by a power failure or a negation of the fault signal.
	Bit 30 (Write data sequence)–Set when write data is not received following a write command. Cleared by a power failure or by the negation of the fault signal.
	Bit 29 (Unexpected read data)–Set when the data read is received and not expected. Cleared by a power failure or the negation of the fault signal.
	Bit 28 (Must be zero).
	Bit 27 (Multiple transmitter)–Set when the source identifier field (ID) of the SBI is different from that transmitted by the MBA while the MBA is transmitting information on the SBI. Cleared by a power failure or by the negation of the fault signal.

Bit	Function
	Bit 26 (Transmit fault) – Set when an SBI fault is detected on the second cycle after the MBA transmits information to the SBI. Cleared by a power failure or by the negation of the fault signal.
25	MBZ (Must be zero).
24	Not used.
23	ADAPT PWR DOWN (Adapter powerdown) – Set when the MBA receives an ac low-voltage condition by the assertion of the AC LO signal and results in a program interrupt. Cleared by the assertion of the INIT, UNJAM, DC LO, or by writing a one to this bit.
22	ADAPT PRW UP (Adapter powerup) – Set when the MBA receives the deassertion of the AC LO signal and results in a processor interrupt. Cleared by the assertion of INIT, UNJAM, DC LO, or by writing a one to this bit.
21	OVER TEMP (Over temperature) – Must be zero.
20:8	MBZ (Must be zero).
7:0	ADAPT CODE (Adapter code) – A unique code that identifies the MBA (00100000).

Computer Interconnect Subsystem Functions

The computer interconnect (CI) subsystem is a high-speed, dual-path data bus and adapter used to locally connect medium- and large-VAX processors, and intelligent mass storage servers into a VAXcluster configuration. The CI750 adapter is used with the VAX-11/750 processor and the CI780 adapter is used with the VAX-11/780 series processors and with the VAX 8600 processor. The adapters mount in the processor backplane and connect to the CI bus, which consists of four coaxial cables: one transmit and one receive cable for each path. Data can be transferred simultaneously on each of the two paths at a rate of 70 Mbits per second. The cables from the CI adapter in the processor connect to the SC008 star coupler unit, which forms the cable termination and distributes the information to the nodes in VAXcluster network. A typical VAXcluster configuration is shown in figure 11-46.

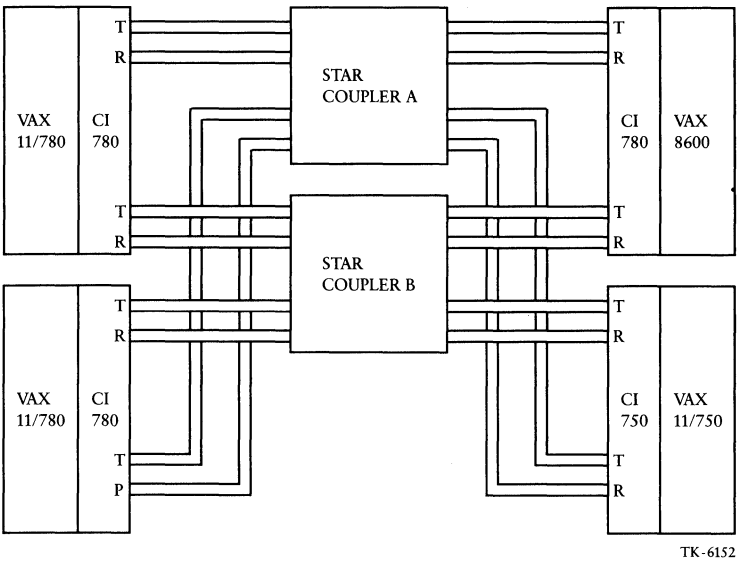


Figure 11-46 • CI Adapter VAXcluster Configuration

- CI ADAPTER OPERATION

The CI adapter is an intelligent high-performance interface that operates as a buffered communications port. It transmits information using a queue structure provided with the VAX/VMS operating system. Messages and data are transferred in blocks between the host system's memory and other nodes within the VAXcluster. The CI adapter provides the data buffering, address translation, and serial encoding and decoding functions to reduce the software normally required to perform these functions. The CI adapters used with the intelligent mass storage subsystems are included with the subsystem logic. Figure 11-47 shows the hardware functions that are included with the interface.

- LINK INTERFACE

The link interface module connects to four communication lines: path A transmit and receives line, and path B transmit and receive lines. Each line is a coaxial cable and a pair of lines form a high-speed communications bus to the VAXcluster configuration. The module is functionally separated into a transmit channel and a receive channel that share a cyclic redundancy check (CRC) function. The link interface logic can serve both paths simultaneously; however, information can be tranferred through only one CI path at a time because of the common CRC logic. This logic generates four CRC

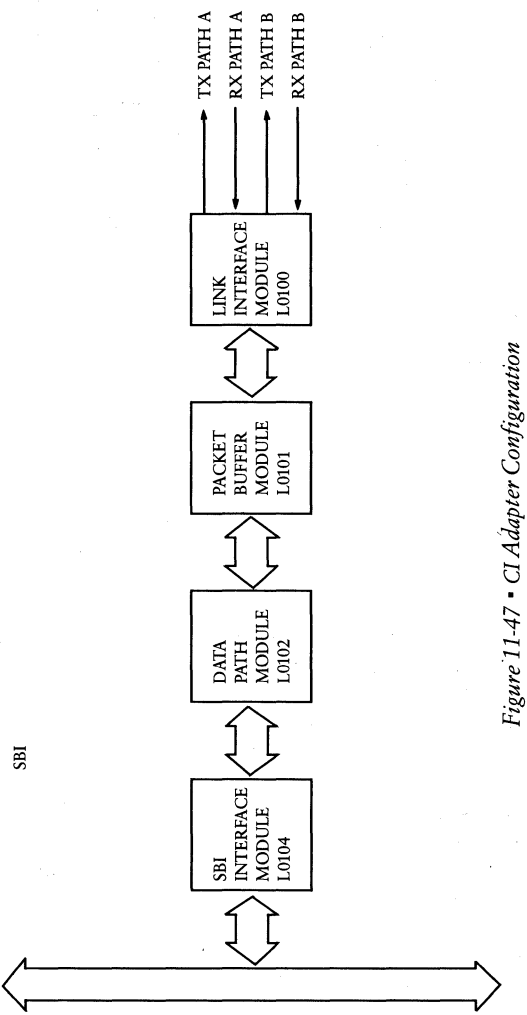


Figure 11-47 ▪ CI Adapter Configuration

bytes that are appended to the data packet and are used for error checking at the packet destination. The link transmitter converts the data packet information to a serial format, and the Manchester encoder combines the serial data with a bit-rate clock to produce a modulated carrier for transmission through the CI bus.

- DATA PACKET BUFFER

  The data packet buffer provides the buffering of the data packets transferred through the CI interconnect. There are two transmit buffers and two receive buffers, each with a storage capacity of 1 Kbyte. The transmit buffers receive data from the SBI through the data path logic for transfer to the line. The receive buffers receive data from the line through the link interface and transfer the information to the SBI. The buffer contains a 3-Kbyte RAM/PROM that stores the microcode to control and regulate the operations of the CI adapter.

- DATA PATH

  The data path converts the SBI's data packet information from longwords to bytes for transmission through the lines and converts bytes of information from the lines to longwords for transmission to the SBI. The data from the SBI is loaded as a longword into a 32-bit transmit register and the register transfers the information as four bytes. The received data bytes are assembled in the 32-bit receive register and transferred as a longword to the SBI. The data path module contains a 256 by 32-bit RAM used to store software status and registers associated with the port software architecture. A 256 by 16-bit RAM provides storage for the virtual circuit descripter table used to store CI node parameters, and an arithmetic logic unit (ALU) is included to perform general purpose arithmetic and logical operations.

- CI REGISTERS

  The CI adapters contain the following four registers that can be accessed by the software for maintenance purposes.

  - Configuration register
  - Port maintenance control/status register
  - Maintenance address register
  - Maintenance data register

Configuration Register–The configuration register (CNFGR) is a read-write register that contains SBI faults bits, port status bits, error bits, and the adapter code for the CI interface. Only a longword can be written into this register and information is read from the register as a byte, word, or longword reference. Any other access mode results in an error confirmation.

The format of the register information for both the CI750 and CI780 adapters is shown in figure 11-48.

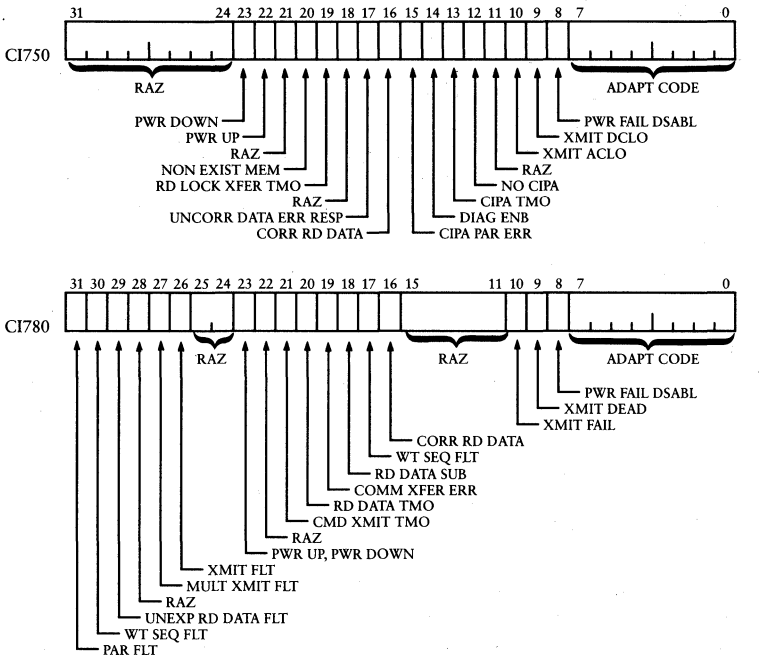


Figure 11-48 ▪ Configuration Register Format

Bit	Function
31:24	RAZ (Read as zero) – For the CI750 adapter.
31	PAR FLT (Parity fault) – Set to indicate that the CI has detected an SBI parity error in the CI780 adapter.
30	WR SEQ FLT (Write sequence fault) – Set when the CI receives a write-mask or interlock-write-mask command that is not immediately followed by the expected write data in the CI780 adapter.
29	UEXP RD DATA FLT (Unexpected read data fault) – Set when the CI adapter received read data and had not issued a read-mask command, extended-read-mask command, or interlock-read-mask command in the CI780 adapter.

Bit	Function
28	RAZ (Read as zero).
27	MULT XMIT FLT (Multiple transmitter fault) – Set when the CI adapter is transmitting information to the SBI and the source/destination (ID) bits transmitted are not the same as the ID bits received in the CI780 adapter.
26	XMIT FLT (Transmit fault) – Set when the CI780 adapter asserts the SBI fault line.
25:24	RAZ (Read as zero) – For the CI780 adapter.
23	PWR DOWN (Powerdown) – Set by the microcode control or by the assertion of the AC LO signal when the CI adapter is in the uninitialized state to indicate that it has powered down; cleared by writing a one to this bit or when the powerup (bit 22) is set.
22	PWR UP (Powerup) – Set by the negation of the AC LO signal to indicate that the CI has powered up; cleared by writing a one to this bit or when the powerdown (bit 23) is set.
21	RAZ (Read as zero).
20	CMD XMIT TMO (Command transmit timeout) – Set when a confirmation is not received within 512 cycles or 2,048 SBI cycles for a SBI command issued by the CI780 adapter.
20	NON EXIST MEM (Nonexistent memory) – Set to indicate that no response was received from a CMI operation initiated by the CI750 adapter.
19	RD LOCK XFER TMO (Read lock transfer timeout) – Set to indicate that a CMI read-lock function could not be initiated by the CI750 adapter for a duration of 1,024 B-clock cycles.
19	RD DATA TMO (Read data timeout) – Set when the CI780 adapter initiates an SBI read command and does not receive data within 512 SBI cycles (102.4 microseconds).
18	RAZ (Read as zero) – For the CI750 adapter.
18	CMD XFER ERR (Command transfer error) – Set when the CI780 adapter receives an error confirmation for an SBI command initiated by the CU adapter.
17	UNCORR DATA ERR RESP (Uncorrectable data error response) – Set to indicate that the response to a CMI operation, issued by the CI750 adapter, contained an uncorrectable data error tag.
17	RD DATA SUB (Read data substitute) – Set when the CI780 adapter response to a read command contains a read data substitute tag.

Bit	Function
16	CORR RD DATA (Corrected read data) – Set when the CI adapter response to a read command contains a corrected read data tag.
15:11	RAZ (Read as zero) – For the CI780 adapter.
15	CIPA PAR ERR (CIPA parity error) – Set when a parity error is detected on the CIPA bus in the CI750 adapter.
14	DIAG ENB (Diagnostic enable) – Set to enable access to the internal diagnostic mode registers in the CI750 adapter.
13	CIPA TMO (CIPA timeout) – Set when an unsolicited CMI bus read or write command is not completed within 10 microseconds in the CI750 adapter.
12	NO CIPA – Cleared to indicate that the CIPA is present, powered up, and initialized in the CI750 adapter.
11	RAZ (Read as zero) – For the CI750 adapter.
10	XMIT FAIL (Transmit fail) – Set by microcode and is the input to the SBI fail driver in the CI780 adapter.
10	XMIT ACLO (Transmit ac low) – Set as an input to the UNIBUS ACLO driver on the CCI module in the CI750 adapter.
9	XMIT DEAD (Transmit dead) – Set by microcode and is an input to the SBI dead driver in the CI780 adapter.
9	XMIT DCLO (Transmit dc low) – Set as an input to the UNIBUS DCLO driver in the CI750 adapter.
8	PWR FAIL DSABL (Power fail disable) – Set to disable the fail or dead indicators to the SBI on the CI780 adapter. Set to disable the AC LO or DC LO signals from being sent to the CMI on the CI750 adapter.
7:0	ADAPT CODE (Adapter code) – A hexadecimal value (38) assigned to the CI adapter.

Port Maintenance Control/Status Register – The port maintenance control/status register (PMCSR) is a read-write register that contains the port hardware error flags and the interrupt and port initialization control bits. The format of the information in the register is shown in figure 11-49.

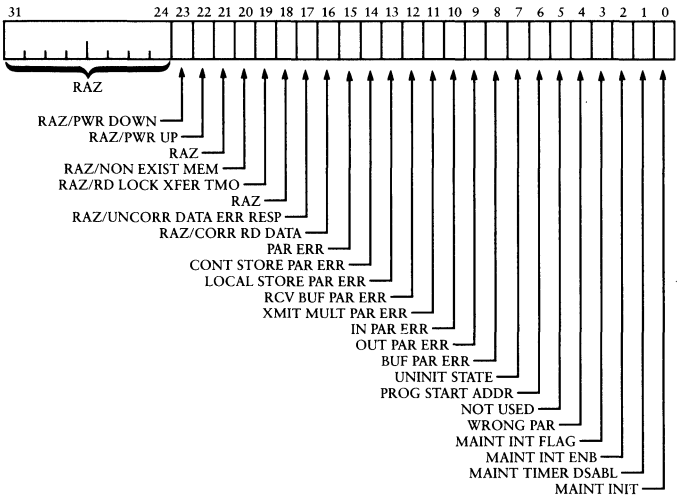


Figure 11-49 • Port Maintenance Control/Status Register Format

Bit	Function
31:16	RAZ (Read as zero) – For the CI780 adapter.
31:24	RAZ (Read as zero) – For the CI750 adapter.
23	PWR DOWN (Powerdown) – Set to indicate that the CI750 adapter is powering down.
22	PWR UP (Powerup) – Set to indicate that the CI750 adapter is powering up.
21	RAZ (Read as zero) – For the CI750 adapter.
20	NON EXIST MEM (Non-existent memory) – Set when no response is received for a CI750-adapter-initiated CMI bus operation.
19	RD LOCK XFER TMO (Readlock transfer timeout) – Set when the CI750 adapter cannot initiate a CMI read-lock function for 1,024 B-clock cycles.
18	RAZ (Read as zero) – For the CI750 adapter.
17	UNCORR DATA ERR RESP (Uncorrectable data error response) – Set when the CI750 adapter initiates a CMI bus operation and the response contains an uncorrectable data error tag.

Bit	Function
16	CORR RD DATA (Corrected read data) – Set when the CI750 adapter initiates a read command and the response contains a corrected read data tag.
15	PAR ERR (Parity error) – Set when a parity error is detected on the CIPA bus of the CI750 adapter and if any of bits 8 through 14 are set in the CI780 adapter.
14	CNTL STORE PAR ERR (Control store parity error) – Set by microcode when a parity error is detected in the control store RAM or PROM. Not used when the control store is accessed from the SBI (CI780 adapter) or from the CMI (CI750 adapter).
13	LOCAL STORE PAR ERR (Local store parity error) – Set by microcode when a parity error is detected in the local store or virtual circuit descriptor table RAMs. Not used when the local store is accessed from the SBI (CI780 adapter) or from the CMI (CI750 adapter).
12	RCV BUF PAR ERR (Receive buffer parity error) – Set when a parity error is detected while reading a receive buffer or transmit buffer of the packet buffer module.
11	XMIT MULT PAR ERR (Transmit multiple parity error) – Set when a parity error is detected in the data being transmitted.
10	IN PAR ERR (Input parity error) – Set when a parity error is detected on a data transfer from the SBI to the CI780 adapter or from the CMI to the CI750 adapter.
9	OUT PAR ERR (Output parity error) – Set when a parity error is detected during a data transfer from the CI780 adapter to the SBI or from the CI750 adapter to the CMI.
8	XMIT BUF PAR ERR (Transmit buffer parity error) – Set when a parity error is detected on the link module transmit buffer of the CI780 adapter or on the packet buffer module of the CI750 adapter during transmission.
7	UNINIT STATE (Uninitialized state) – Set when the microcode has stopped operation and the CI adapter is not initialized.
6	PROG START ADDR (Programmable starting address) – Set to start the microcode at the address selected by the maintenance address register (MADR), when bit 0 is set in the port initialize control register (PICR), or when a bootstrap sequence timeout occurs. When cleared, the microcode will start at address 000.
5	Not used.

Bit	Function
4	WRONG PAR (Wrong parity) – Set to check the even parity on the data path module.
3	MAIN INT FLAG (Maintenance interrupt flag) – Set by a condition in the CI adapter that causes an interrupt to occur when the contents of the PSR register are valid. Allows the diagnostic program to operate the adapter with the interrupts to the SBI (CI780 adapter) or the CMI (CI750 adapter) disabled.
2	MAINT INT ENB (Maintenance interrupt enable) – Set by writing a one to this bit or by the DC LO signal to enable the interrupts. Cleared by writing a zero to this bit, by setting the MAINT INIT (bit 0), or by the SBI unjam signal.
1	MAINT TIMER DSABL (Maintenance timer disabled) – Set to disable the boot and sanity timers in the CI adapter, thereby preventing an interrupt.
0	MAINT INIT (Maintenance initialize) – Set to generate an initialize signal to clear the CI adapter port errors. This bit is always read as a zero. The CI adapter is uninitialized and the MAINT ENB (bit 2) is cleared.

Maintenance Address Register – The maintenance address register (MADR) is a read-write register that is accessible when the CI adapter is in the uninitialized state. The address loaded into the MADR is used as a pointer to locations in the control store RAM. The format of the register information is shown in figure 11-50.

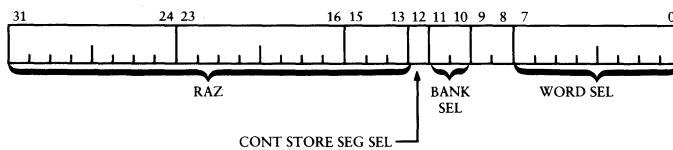


Figure 11-50 • Maintenance Address Register Format

Bit	Function
31:13	RAZ (Read as zero).
12	CNTL STORE SEG SEL (Control store segment select) – Selects a segment of the control store word as follows: 0 = bits 31:0 1 = bits 47:32
11:10	BANK SEL (Bank Select) – A binary value that selects a bank within the control store as follows:

Value	Bank (hexadecimal)
0	0 (000 - 3FF)
1	1 (400 - 7FF)
2	2 (800 - BFF)
3	3 (C00 - FFF reserved)

Bit	Function
9:0	WORD SEL (Word select) – A binary number used to access a word within the bank selected.

Maintenance Data Register – The maintenance data register (MDATR) is a read-write register used to access the contents of the control store location selected by the MADR and is used to initially load the microcode. The register is valid only when the CI adapter is in the uninitialized state. The format for the MDATR is shown in figure 11-51.

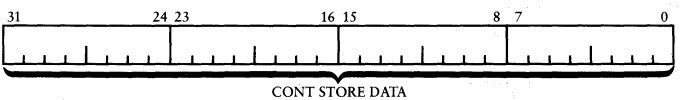


Figure 11-51 ▪ Maintenance Data Register Format

Bit	Function
13:0	CONT STORE DATA (Control store data) – The contents of the control store location specified by the MADR.

Port Status Register–The port status register (PSR) is a read-only register that contains status flags to indicate the cause of an interrupt generated by the CI interface. Figure 11-52 shows the format of the status information in the register.

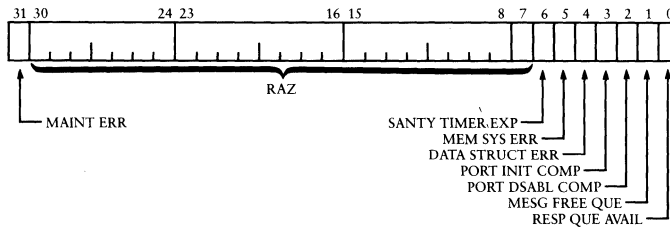


Figure 11-52 • Port Status Register Format

Bit	Function
31	MAINT ERR (Maintenance error)–Set to indicate that the CI adapter port has detected an internal hardware failure. When set the CI enters the unitialized state, the microcode is halted and an interrupt request is generated. The error is determined by the information in the PMSCR or CNFGR.
30:7	RAZ (Read as zero).
6	SANTY TIMER EXPIR (Sanity timer expiration)–Set when the sanity or boot timer has expired and the CI adapter port has entered the initialized maintenance state.
5	MEM SYS ERR (Memory system error)–Set when the CI adapter has detected uncorrectable data or a nonexistent memory error while accessing SBI memory (CI780 adapter) or CMI memory (CI750 adapter). The PFAR contains additional information related to this error.
4	DATA STRUCT ERR (Data structure error)–Set when an error is detected in the port data structure. The PFAR contains additional information related to the error.
3	PORT INIT COMPL (Port initialization complete)–Set when the port has completed internal initialization and is in the disabled or disabled/maintenance state.

Bit	Function
2	PORT DSABL COMPL (Port disable complete) – Set when the port ceases to process the command queues and to respond to incoming transmissions except for the maintenance class. The port is in the disabled or disabled/maintenance state.
1	MESG FREE QUE (Message free queue) – Set when the port attempts to access an entry from a message free queue.
0	RESP QUE AVAIL (Response queue available) – Set when the port has inserted an entry in an empty response queue.

Port Failing Address Register – The port failing address register (PFAR) is a read-only register that contains the memory address at which a failure occurred after a memory system or data structure error or after a response that includes a buffer-memory-system error status. This register may contain the failing address, an address in the same page as the failing address, or an address in part of the data structure. Figure 11-53 shows the format of the information in the register. The register is used by the port driver software when the microcode is operating.

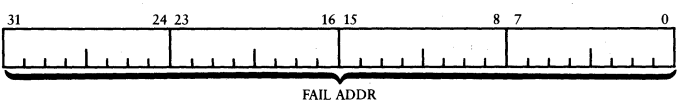


Figure 11-53 ▪ Port Failing-Address Register Format

Bit	Function
31:0	FAIL ADDR (Failing address) – Contains a physical address for memory system errors (MSE) and buffer memory system errors. Contains a virtual address or offset value for data system errors (DSE). The PESR may contain additional information related to the contents of this register for DSE interrupts.

Port Error Status Register–The port error status register (PESR) is a read-only register that identifies the type of error that caused the data structure error. This register is used by the port driver software and is valid when the microcode is operating. The register format is shown in figure 11-54. Error indications are listed in table 11-12.

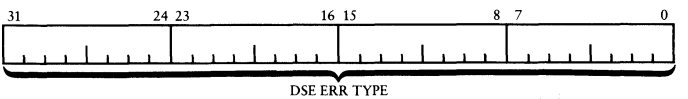


Figure 11-54 • Port Error Status Register Format

Bit	Function
31:0	DSE ERR TYPE (Data structure error type)–A hexadecimal code specifying the data structure error.

TABLE 11-12 • PASR Error Codes

Code	Register protocol violation	PFAR Contents*
1	Illegal system virtual–address format	1
2	Nonexistent system virtual address	1
3	Invalid system PTE	1
4	Invalid buffer PTE	2
5	Nonexistent system global virtual address	1
6	Nonexistent buffer global virtual address	2
7	Invalid system global PTE	1
8	Invalid buffer global PTE	2
9	Invalid system global PTE mapping	1
A	Invalid buffer global PTE mapping	1
B	Queue interlock retry failure	3

Code	Register protocol violation	PFAR Contents*
C	Illegal queue offset alignment	3
D	Illegal PQB format	4
E	Register protocol violation	5

*1 = Virtual address
2 = PTE virtual address
3 = Queue head virtual address
4 = PQB field offset in bytes
5 = Register byte offset from device base address

Port Parameter Register–The port parameter register (PPR), a read-only register, is loaded by the microcode during the CI adapter initialization process and is not valid in the uninitialized state. The register indicates the number selected by the CI adapter module switches. Figure 11-55 shows the register format.

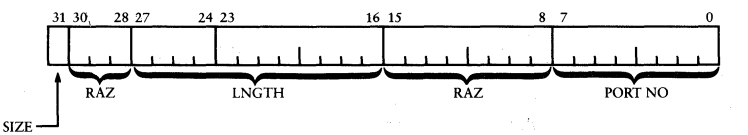


Figure 11-55 ▪ Port Parameter Register Format

Bit	Function
31	SIZE–Initialized to zero and set to indicate that the 16 (maximum) CI adapter nodes are contained in the system.
30:28	RAZ (Read as zero).
27:16	LNGTH (Length)–Indicates the size of the internal buffers and is preset to 3F9 (hexadecimal).
15:8	RAZ (Read as zero).
7:0	PORT NO–Indicates the hexadecimal value of the CI adapter node.

Port Initialize Control Register–The port initialize control register (PICR) is a write-only register that is initialized by the CI adapter software driver to start the execution of the microcode. The format of the register is shown in figure 11-56.

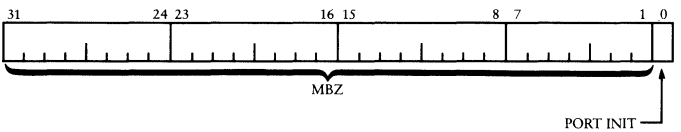


Figure 11-56 • Port Initialize Control Register Format

FIGURE 11-56 • Port Initialize Control Register

Bit	Function
31:1	MBZ (Must be zero).
0	PORT INIT (Port initialize)–Set to initialize the CI adapter.

• PACKET FORMATS

Information is transferred between the host memory and other nodes connected to the CI adapter in blocks consisting of sequential bytes arranged in packets. The two types of packets used, the information packet and the acknowledgement packet, are shown in figure 11-57. Each packet consists of synchronization bytes, packet description bytes, source and destination information, the message body, and the trailer bytes.

Information Packet–The information packet can transfer messages or data through the CI adapter, and some of the control information is appended by the CI adapter. The first five bytes provide bit synchronization and include four bytes written with alternate zeros and ones to activate the carrier detect circuits and to synchronize the Manchester decoder. The last byte is the character synchronization, which indicates the start of packet information. The packet-type/length (high) byte specifies an information packet type or an acknowledge packet type and contains the upper four bits of the packet length word. Information packets are variable in length up to 1 Kbyte. The packet length (low) byte contains the low eight bits of the packet length. Two destination bytes (true and complement) provide an 8-bit address of the CI node to which the packet is to be transmitted. The

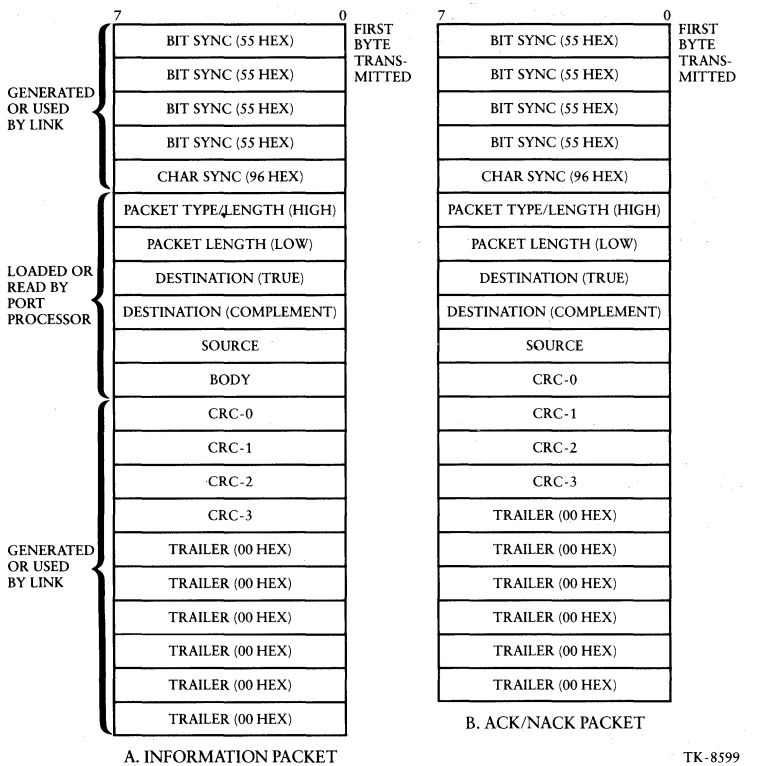


Figure 11-57 ▪ CI Subsystem Packet Configuration

true byte is the actual address value of the node that is to receive the information and the complement byte is the logical complement of the true address. The source byte is an 8-bit address of the node that is sending the message. The body bytes follow and contain data and port-processed protocol. The body can be as many bytes as required, provided that the total packet size does not exceed 1 Kbyte. Following the body bytes are four cyclic redundancy check bits (CRC) that provide a unique code representing the information in the packet to ensure that information is not lost or in error during the transmission. The trailer consists of six bytes written entirely with zeros to identify the end of a transmission and to prevent the node from disengaging before the last data bytes are processed.

Acknowledge–Negative/Acknowledge Packet–The acknowledge-negative/acknowledge packet is sent by the receiving node to inform the transmit-

ting node that the packet arrived without data loss or data collisions caused by simultaneous data transmissions. If the receiving node successfully accepts the packet, an acknowledge packet is returned to the sender indicating a successful transmission. If the receive buffers in the CI contain information when the information arrives, the negative acknowledge packet is sent to inform the transmitting node that the packet was successfully received but not accepted and the transmitter is required to retransmit the packet.

DR32 Device Interconnect Subsystem Functions

The DR32 device interconnect (DDI) is a microprocessor-controlled, high-performance, I/O subsystem that allows users' devices or medium- and large-VAX processors to be connected through the DDI 32-bit parallel data bus. The DR750 adapter is used with the VAX-11/750 processor and connects to the CMI bus. The DR780 adapter is used with the VAX-11/780 series processors and with the VAX 8600 processor and connects to the SBI bus. Direct memory access (DMA) communications is established between the processor memories or between processor memory and devices, and data can be transferred at rates of up to 6.67 Mbytes per second.

The DDI subsystem is software supported by the VAX/VMS operating system with a simple, easy-to-use I/O driver and a library of high-level language support routines. Data verification is performed by parity checking on both control and data transfers. Error detection and logging, address range checking on data transfers and online diagnosis are included with the DDI subsystem.

- DDI ADAPTER OPERATION

The DDI adapter consists of four modules and a connecting bus. Figure 11-58 shows the configuration of the DDI adapter modules and the SBI and I/O bus connections. The DDI adapter consists of an SBI control module (DSC) that connects to the SBI bus and transfers the information between adapter and SBI, performs the address translation, and provides the control function for sequencing the information to and from the SBI bus. The control board (DCB) performs the internal control function to sequence the information internally within the adapter. The microprocessor and microcode (DUP) module perform the intelligent operations required to format and control the information for DMA transfer, thereby eliminating the operations normally performed by the processor. The silo module (DSM) assembles the data, address, and control bytes into packets during the reception and transmission.

The 32-bit DDI bus is a synchronous path for transferring the data to and from the interconnecting systems or devices. The data transfer rate is

synchronized by an internal clock or by an external clock provided by the device. Transmission rates depend on the configuration of the system. The characteristics of the DDI bus are defined by the *Digital DR32* specification.

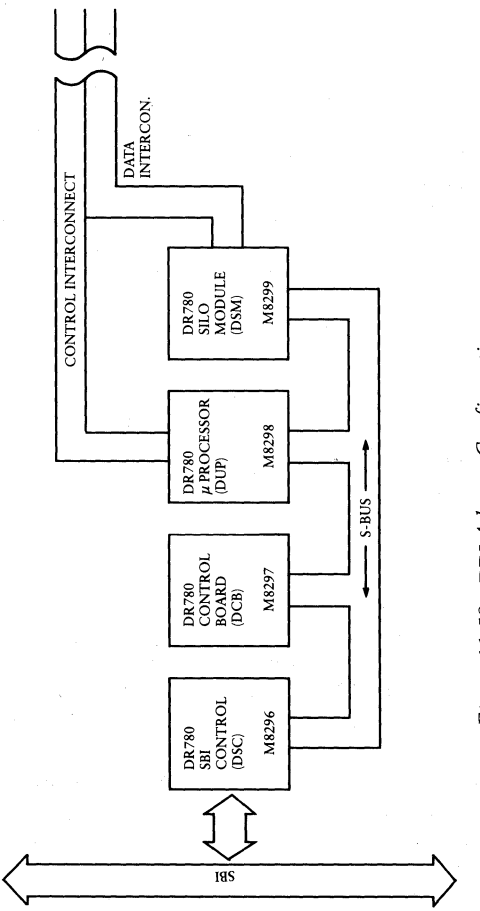


Figure 11-58 ▪ DDI Adapter Configuration

- DDI SIGNAL LINES

Figure 11-59 shows the signal lines that connect to the device or processor. The DDI is a bidirectional bus for transferring data and control signals. Control signals are asynchronous and interlocked, and data transfers are synchronized with clock signals. A processor or a device can initiate the transfers. The DDI provides point-to-point connection and consists of 32 parallel data lines and 8 control lines. The control lines allow the DDI to address up to 256 individual registers. These registers can be used for buffer descriptor commands or status information of the user process.

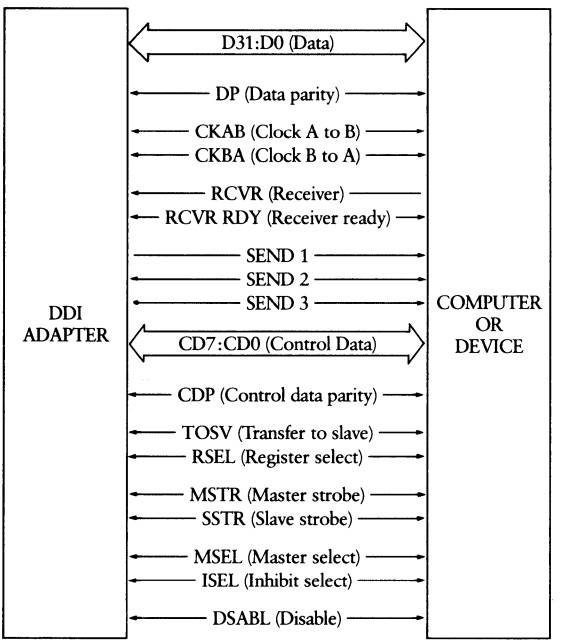


Figure 11-59 • DDI Signal Lines

Data transfers can be synchronized with the clock that is included with the DDI adapter or with an external clock provided by the user. Transfer rates from 0.156 to 6.67 Mbytes per second can be selected by program. Table 11-13 lists the signal lines and defines their functions.

TABLE 11-13 ▪ DDI Signal Line Functions

Line	Description
D31:D0	Data – Bidirectional lines that contain the data for all transactions.
DP	Data parity – A bidirectional line that indicates the parity of the information on the D31:D0 lines.
CKAB	Clock A to B – A unidirectional line used to synchronize the data and control information from device A to B.
CKBA	Clock B to A – A unidirectional line used to send data from device B to A.
RCVR	Receive – A bidirectional line asserted by the clock signals on lines CKAB or CKBA when the receiver is ready to receive data.
RCVR READY	Receiver ready – A bidirectional line asserted by the clock signal on the CKAB or CKBA line only if the RCVR line is asserted.
SEND 1, 2, 3	Three bidirectional lines from the sender that contain a binary code for the type of function and the information to be transferred, as follows:

			Data Positions	
Code	**Bytes**	**Function**	**Send**	**Data**
0	0	0	0	none
1	0	1	0	none
2	0	1	0	none
3	0	1	0	none
4	4	1	1	DATA 31:0
5	1	1	1	DATA 7:0
6	2	1	1	DATA 15:0
7	3	1	1	DATA 23:0

CD07:CD0	Control data – Bidirectional lines asserted by the master for a write operation or by a slave for a control interconnect read operation.
CDP	Control data parity – A bidirectional line asserted as the parity bit for the CD07:CD0 data lines.
TOSV	Transfer to slave – A bidirectional line asserted by the master to indicate that the direction of transfer is from the master to the slave.

Line	Description
RSEL	Register select – A bidirectional line asserted to indicate that the information on CD07:CD0 is to be loaded into the address register of the device if the TOSV line is asserted. The contents of the address register are to be placed on CD07:CD0 if the TOSV line is not asserted. It is also asserted by the slave to indicate that the device cannot complete the transfer.
MSTR	Master strobe – A bidirectional line asserted by the master to indicate that the information on the CD07:CD0, CDP, TOSV, and RSEL lines are stable.
SSTR	Slave strobe – A bidirectional line asserted by the slave to indicate that data has been received during a write sequence and data has been sent during a read sequence.
MSEL	Master select – A unidirectional line asserted by the device or processor selected as the arbitrator. When it is not asserted, the remaining processor or device assumes the arbitration function.
ISEL	Inhibit select – A bidirectional line asserted by either the processor or device to indicate to the arbitrator that the current level on the MSEL line should be maintained.
DSABL	Disable – A unidirectional line asserted by the processor or device to direct the receiver to ignore the contents of the DDI lines.

- COMMAND CHAINING

The DDI permits a series of commands to be executed without intervention of the software for each command. Commands can be chained together in a command queue and executed in succession. After the command sequence is initiated, the process continues until the transfer is halted by a command packet that contains a *halt* command or by an error. The DDI option can perform dynamic command chaining so that commands can be added or deleted from the command queue while the DDI adapter is operating. Four of the features provided by command chaining are as follows:

- Enables commands to be continuously fetched from main memory by the microprocessor. The microprocessor requests command information and transfers data continuously from the virtual memory of the user's process without program intervention.

- Enables direct communication through queues between the processor and the user's process. This allows the process to directly control the DDI adapter and the same I/O driver software to be used in a wide range of applications. Since the user's process builds command packets, the I/O drivers perform only privileged tasks for the user process, such as supplying the DDI adapter with the address range of operation, fielding interrupts, and aborting the current operation.
- Includes dynamic memory mapping to allow continuous data transfers of arbitrary length to be performed without reloading address translation map registers. Data transfers are specified as virtual addresses that the DDI adapter dynamically translates to physical address locations, thereby eliminating the need for mapping of registers. The DDI buffer size is limited only by the amount of physical memory available. The DDI uses the same page-table entries as the CPU.
- Allows concurrent data and command transfer by the use of separate and independent data and control lines. The subsequent command sequence can be established while a current data transfer is being completed.

- DATA CHAINING

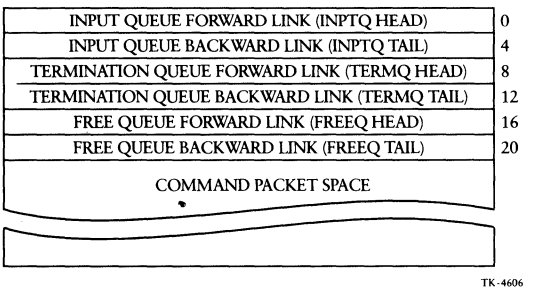
Command packets can specify data chaining, allowing several main memory buffers to appear to the device as one large buffer and the transfers to be a continuous stream of data. Chained buffers can be aligned by byte and can be any length greater than 32 bits up to the maximum physical memory size. The device or host processor can initiate data transfers in a random access mode. This mode is used when two DDI subsystems are connected to form a processor-to-processor link. Random access consists of data transfers to or from the memory without notification of the processor and can be discontinued by either specifying a command packet with random access disabled or by an abort from either the controlling process or the device.

If a power failure occurs on the DDI but not on the system, the I/O driver aborts the active data transfer and returns a status code in the I/O status block. If a system power failure occurs, the driver completes the active data transfer when power is recovered and returns a powerfail status code.

The DDI can interrupt the I/O driver when an abort, a powerdown, or a powerup sequence occurs, when an unsolicited control message has been sent to the DDI, when the DDI enters the halt state, or by using a command packet that specifies an unconditional interrupt.

- PROGRAMMING INTERFACE

Direct communication between the controlling process and the DDI is enabled by the commands from user-constructed command packets located in main memory. Command packets contain commands that specify the direction of transfer or the messages to be sent. The controlling process builds command packets and manipulates the three queues, using the four instructions. Figure 11-60 shows the contents of a command packet.



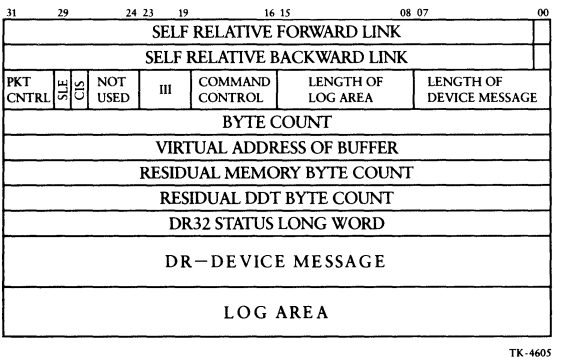


Figure 11-60 • DDI Subsystem Packet Configuration

The DDI subsystem is supported by a device driver and a high-level language procedure library of support routines. After the driver initializes the DDI, application programs can communicate directly by inserting command packets into queues. This direct link between the application program and the DDI provides faster communication by eliminating the use of the I/O driver.

The DDI is accessed by the queue I/O (QIO) driver and a set of support routines. Application programs are used to build and insert command packets into the DDI queues. The set of support routines provides procedures for building and manipulating command packets and performing housekeeping functions such as maintaining command memory. The DDI program interface contains input, termination, and free queues that control the flow of command packets to and from the DDI interface. The application program builds a command packet and inserts it onto the input queue; the DDI adapter removes the packet, executes the specified command, and inserts the command packet into the termination queue. Unsolicited input, such as a control message from the user device, is placed in packets removed from the free queue and inserted into the termination queue for later processing. Figure 11-61 shows the flow of command packets as they are moved to the three queues.

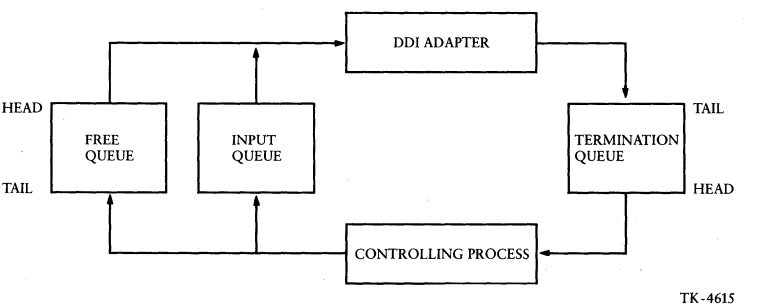


Figure 11-61 ▪ DDI Command Packet Flow

The application program interfaces with memory through the command block and the buffer block. The command block contains the headers for the three queues that provide the communication path between the DDI and the application program and space for the command packets to be built. The buffer block defines the area of memory that is accessible to the DDI for the transfer of data between the user device and the DDI. If a command packet is inserted into an empty input queue, the application program must notify the DDI adapter; the packets are processed until the queue is empty.

Unsolicited control messages from devices can be accepted by the DDI but are not synchronized to the application program command flow. The application program inserts a number of empty command packets into the free

queue to allow the external device to transmit the control messages. If a control message is received from the device, the DDI removes an empty command packet from the head of the free queue, fills the device message field of this packet with the control message, and inserts the packet into the bottom of the termination queue. The device message field in this command packet must be able to contain the entire message, or a length error will occur. The application program then removes the packet from this queue.

- DDI ADAPTER REGISTERS

The DDI adapter contains several registers that are accessible to the program or to the user through the console terminal. These DCR control registers and the user-control registers are accessible by program during normal operation of the DDI adapter. The remaining registers can be accessed only when the DDI adapter is in the halt state.

- DEVICE CONTROL REGISTER

The device control register (DCR) is a read-write register that provides control and status information during read or write operations. Figure 11-62 shows the register format for the read operation and figure 11-63 shows the register format for the write operations.

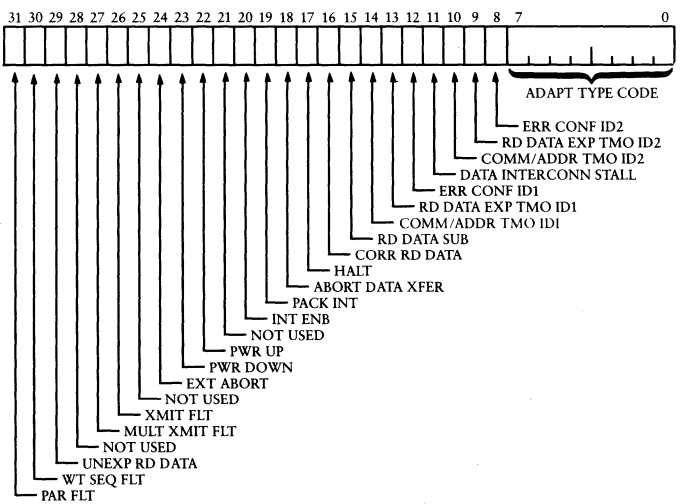


Figure 11-62 • Device-Control Read Register Format

Bit	Function
31	PAR FLT (Parity fault) – Set when a parity fault has been detected during a data transfer.
30	WT SEQ FLT (Write sequence fault) – Set when a fault has occurred during the write sequence.
29	UNEXP RD DATA (Unexpected read data) – Set when the DDI adapter has received a read command before it had completed the preceding transaction.
28	Not used.
27	MULT XMIT FLT (Multiple transmitter fault) – Set when the identifier field (ID) of the SBI is different from the ID transmitted by the DDI adapter.
26	XMIT FLT (Transmit fault) – Set when an SBI fault is detected after the DDI adapter transmits information to the SBI bus.
25	Not used.
24	EXT ABORT (External abort) – Set when the DDI adapter operation is halted by an external function.
23	PWR DOWN (Powerdown) – Set when the DDI adapter is powered down.
22	PWR UP (Powerup) – Set when the DDI adapter has powered up to request a processor interrupt.
21	Not used.
20	INT ENB (Interrupt enable) – Set to allow the DDI adapter to initiate an interrupt request.
19	PACK INT (Packet interrupt) – Set to initiate an interrupt request when instructed by the information in a data or command packet.
18	ABORT DATA XFER (Abort data transfer) – Set to initiate an abort sequence to stop data transfers.
17	HALT.
16	CORR RD DATA (Corrected read data) – Set when the SBI response to a read command contains a corrected read data mask.
15	RD DATA SUB (Read data substitute) – Set when main memory has an uncorrectable error on a read operation.
14	COMM/ADDR TMO ID1 (Command/address timeout ID1) – Set when time allowed between the initiation of a command or address and the acknowledge of the command has been exceeded.

Bit	Function
13	RD DATA EXP TMO ID1 (Read data expected timeout ID1) – Set when the time allowed between the initiation of a read command and the receipt of the data by the requesting device is exceeded.
12	ERR CONF ID1 (Error confirmation ID1) – Set when main memory cannot perform the command from the DDI adapter.
11	DATA INTRCONN STALL (Data interconnect stall).
10	COMM/ADDR TMO ID2 (Command/address timeout ID2) – Set when the time allowed between the initiation of a command or address and the acknowledgement of the command is exceeded.
9	RD DATA EXP TMO ID2 (Read data expected timeout ID2) – Set when the time allowed between the initiation of a read command and receipt of the data by the requesting device is exceeded.
8	ERR CONF ID2 (Error confirmation ID2) – Set when the DDI receives an error confirmation for a read or write command.
7:0	ADAPT TYPE CODE (Adapter type code) – A hexadecimal value (30) that specifies the type of DDI adapter in the system.

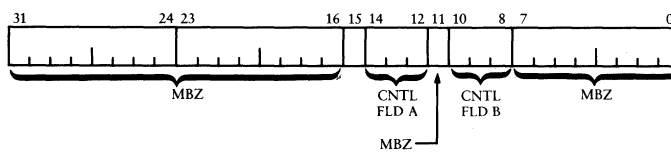


Figure 11-63 • Device-Control Write Register Format

Bit	Function
31:15	MBZ (Must be zero).
14:12	CNTL FLD A (Control field A) – A binary number used to select the following control functions.
	0 = No operation – The DDI adapter ignores the current command and continues to the next command.
	1 = Clear corrected read data – Clear the corrected read data (bit 16).
	2 = Set abort – Used by the I/O driver to set the abort (bit 18) and start the abort sequence.

Bit	Function
	3 = Clear packet interrupt–Clears the packet interrupt (bit 19) to prevent the DDI adapter from interrupting the processor.
	4 = Reset–Resets the DDI adapter to a known state.
	5 = Set out of sequence–Used for diagnostic maintenance.
	6 = Clear out of sequence–Used for diagnostic maintenance.
	7 = No operation–The DDI ignores the current command and continues to the next command.
11	MBZ (must be zero).
10:8	CNTL FLD B (Control field B)–A binary number used to select the following control functions.
	0 = No operation–The DDI adapter ignores the current command and continues to the next command.
	1 = Clear adapter powerup–Clears the powerup (bit 22) to prevent a processor interrupt.
	2 = Clear adapter powerdown–Clears the powerdown (bit 23) to prevent a processor interrupt.
	3 = No operation–The DDI adapter ignores the current command and continues to the next command.
	4 = Clear abort interrupt and read data substitute–Clears the abort (bit 18) and the read data substitute (bit 15).
	5 = Clear interrupt enable–Clears the interrupt enable (bit 20) to prevent the generation of an interrupt request after the command packets have executed.
	6 = Set interrupt enable–Sets the interrupt enable (bit 20) to allow the DDI adapter to initiate an interrupt request.
	7 = Clear halt–Used by the I/O driver to clear the internal halt (bit 17) and allow the DDI adapter to process commands.
7:0	MBZ (Must be zero).

Utility Register—The utility register (UTR) is a read-write register that contains miscellaneous status information from the DDI adapter. Figure 11-64 shows the format of the register information.

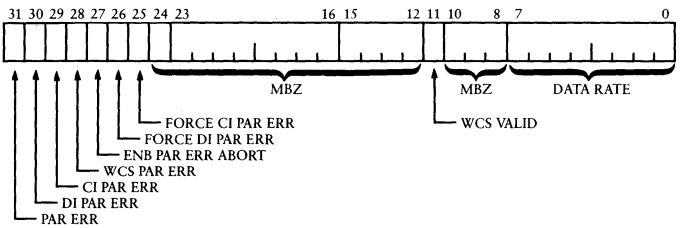


Figure 11-64 • Utility Register Format

Bit	Function
31	PAR ERR (Parity error)—Set when a parity error is detected on the control or data information on the DDI bus, or on the writable control store.
30	DI PAR ERR (Data interconnect parity error)—Set when a parity error is detected on the data interconnect information.
29	CI PAR ERR (Control interconnect parity error)—Set when a parity error is detected on the control interconnect.
28	WCS PAR ERR (Writable control store parity error)—Set when a writable-control-store RAM error is detected.
27	ENB PAR ERR ABORT (Enable parity error abort)—Set to cause the DDI adapter to abort the data transfer and to initiate an interrupt request.
26	FORCE DI PAR ERR (Force a DI parity error)—Set by the diagnostic program to test the operation of the parity circuits for the data interconnect.
25	FORCE CI PAR ERR (Force a CI parity error)—Set by the diagnostic program to test the operation of the parity circuits for the control interconnect.
24:12	MBZ (Must be zero).
11	WCS VALID—Set to allow the microprocessor to operate and cleared to cause the DDI adapter to abort the data transfer.

Bit	Function
10:8	MBZ (Must be zero).
7:0	DATA RATE–Selects the rate at which the data is transferred through the DDI adapter. This number is the two's complement of the count derived from the following equation.
	40 data rate (Mbytes per second) = 256 (data rate in bytes)

WCS Address Register–The writable-control-store address register (WCSAR) is a write-only register that is used to address the WCS memory for loading the microprogram into the microprocessor. The address format of the WCSAR is shown in figure 11-65.

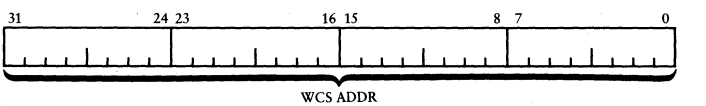


Figure 11-65 ▪ WCS Address Register Format

Bit	Function
31:0	WCS ADDR–Contains the address of the WCS for the location in the WCS memory where the data is to be transferred.

WCS Data Register–The writable-control-store data register (WCSDR) is a write-only register that is used to hold the data for transfer to the WCS memory at the address indicated by the WCSAR. The format of the data is shown in figure 11-66.

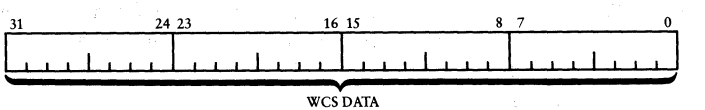


Figure 11-66 ▪ WCS Data Register Format

Bit	Function
31:0	WCS DATA – Contains the data to be loaded into the WCS memory at the address specified by the WCSAR.

Address Register – The address register (ADR) is a read-write register that is loaded by the microprocessor with the physical memory address for the transfer of data in response to a command packet. This register is accessed only when the DDI adapter is in the halt state. Figure 11-67 shows the format of the information in the register.

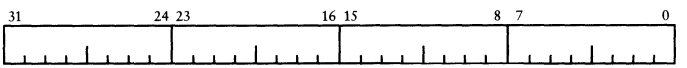


Figure 11-67 • Address Register Format

Bit	Function
31:0	PHYS MEM ADDR – Contains the physical memory address for the transfer of data from the DDI adapter.

SBI Byte Count Register – The SBI byte count (SBIBC) is a read-write register loaded by the microprocessor and used to monitor the number of bytes transferred to the SBI bus. Figure 11-68 shows the register format.

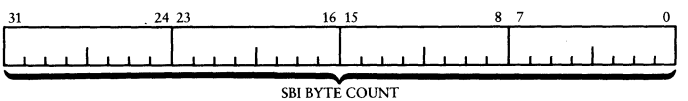


Figure 11-68 • SBI Byte Count Register Format

Bit	Function
31:0	SBI BYTE COUNT – Contains the count of the number of bytes to be transferred from the DDI adapter to the SBI bus during the transaction.

DDI Byte Count Register–The DDI byte count (DDIBC) is a read-write register loaded by the microprocessor and used to monitor the number of bytes transferred to the DDI adapter. Figure 11-69 shows the register format.

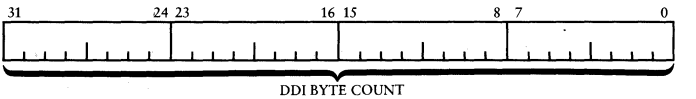


Figure 11-69 ▪ DDI Byte Count Register Format

Bit	Function
13:0	DDI BYTE COUNT–Contains a count value of the number of bytes to be transferred to the DDI adapter during the transaction.

User Control Register–The user control register (UCR) is a write-only register used to enable the microprocessor operation and start executing command packets. Figure 11-70 shows the format of the register information.

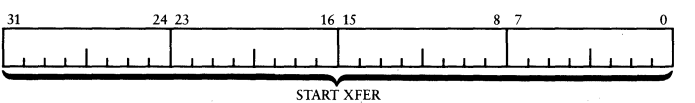


Figure 11-70 ▪ User Control Register Format

Bit	Function
31:0	START XFER (Start transfer)–Set to initiate the execution of the command packets.

Chapter 12 · Mass Storage Subsystems and Devices

Digital offers a complete line of intelligent mass storage subsystems and storage devices designed specifically to operate with VAX processors and VAXcluster architecture. These subsystems and devices allow a user to select the mass storage requirements that best suit the system application and to add storage capabilities when the application requirements increase. The storage subsystems and devices operate with the UNIBUS, MASSBUS, and VAXclusters to provide fast and reliable access to information. The interaction of the VAX processors normally required to access and manage data has been reduced by allocating many of the data tasks to the mass storage subsystem and devices. Refer to the *VAX Systems and Options Catalog* for detailed device descriptions, configuring information, and ordering codes of the devices available.

Digital Storage Architecture (DSA) is the standard for the design and manufacture of storage subsystems and devices and is a framework for an expanding group of mass storage products and intelligent controllers. The DSA provides data reliability and integrity and allows efficient performance and ease of maintenance for all Digital storage subsystems and devices. Within the DSA framework, the disk drives, magnetic tape units, and intelligent controllers are compatible with each other, allowing any of the devices to operate with any intelligent controller. New systems and devices that are developed by Digital can be easily added to the system without the need to replace existing storage devices or to generate new software.

Some of the features provided by the DNA and storage subsystem design are

- Advanced data integrity techniques.
- High-capacity mass storage devices.
- Intelligent controllers that provide error detection and improved throughput.
- Easy migration to new Digital mass storage products.
- Integrated host processor and storage subsystem engineering.
- Dual-access drive capabilities.

▪ Intelligent Mass Storage Controllers

The HSC50 and the UDA50 are intelligent controllers that perform many of the functions required to effectively manage and control the storage and transfer of information. The HSC50 is a special-purpose storage controller that operates as a node within the VAXcluster architecture and coordinates the activities of up to 24 mass storage devices, including disk and magnetic tape drives. The HSC50 controller can serve several host processors connected to the VAXcluster CI bus. The UDA50 controller is designed for single processor VAX systems and connects up to four disk drives to the UNIBUS.

HSC50 Intelligent Mass Storage Server

The HSC50 is an intelligent mass storage controller that operates in a VAXcluster configuration. It allows VAX systems to share data stored in the devices connected to the HSC50 controller. The HSC50 controller, shown in figure 12-1, is contained in a cabinet 106.7 centimeters (42 inches) high, and 53.9 centimeters (21.25 inches) wide. The HSC50 includes a power system and is electrically isolated from the processors on the VAXcluster CI bus. It provides high-performance and fast I/O data throughput using a PDP-11 control processor and high-speed bit-slice microprocessors. An internal high-speed bus permits simultaneous read or write operations to the disk drives at a sustained transfer rate of up to 4 Mbytes per second. The HSC50 offloads all disk management functions from the host processor and provides the independent sharing of the database with the processors on the VAXcluster. The controller stores up to 120 transfer requests and provides buffering for 128 Kbytes of data. Self-contained diagnostics software is included in the HSC50 controller to facilitate maintenance functions. Storage devices that operate with the HSC50 unit are the RA60, RA80, and RA81 disk drives and the TA78 magnetic tape unit.

The *performance specifications* of the HSC50 controller are

- CI port bandwidth: 8.8 Mbytes per second (maximum).
- Disk data-channel bandwidth: 3.125 Mbytes per second (maximum) each channel.
- Internal data bus: 13.3 Mbytes per second (maximum).
- Request processing overhead: 1.6 milliseconds per request.

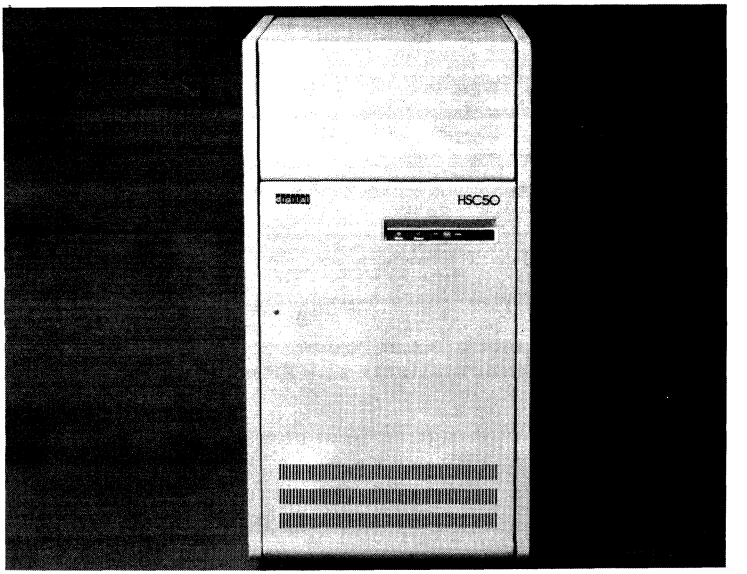


Figure 12-1 • HSC50 Intelligent Mass Storage Controller

UDA50 Intelligent Disk-drive Controller

The UDA50 is an intelligent disk-drive controller that operates with the UNIBUS of the VAX systems and supports up to four disk-drive devices. The UDA50 controller, shown in figure 12-2, is contained on two hex-height modules that are installed into any two adjacent UNIBUS backplane slots. The modules provide the UNIBUS interface and disk control functions for communication between the disk drives and the host processor. Two microprocessors are included to control the routing of data in and out of the memory buffers located on the controller. A command queue stores up to 20 transfer requests from the processors and a speed-matching buffer provides the buffering for up to 52 disk sectors. Blocks of sequential data are transferred to memory by direct memory access (DMA) requests through the UNIBUS. The UDA50 contains diagnostic software for maintenance functions. Fault conditions are indicated by lights on the module and by a register that can be monitored by the host processor. A last-fault register contains the last fault detected and stored in an error register. Up to four disk drives – RA60, RA80, or RA81 disk drives in any combination – can operate with the UDA50 controller. Figure 12-2 shows the UDA50 controller module set.

Figure 12-2 ▪ UDA50 Intelligent Disk-drive Controller

The *performance specifications* for the UDA50 controller are

- Disk data transfer rates: 3.125 Mbytes per second (maximum).
- UNIBUS data transfer rates: 1.2 Mbytes per second (maximum).
- Capacity: 1.8 Gbytes per controller.

▪ DSA Disk-drive Subsystems

The disk-drive subsystems that operate with the HSC50 and UDA50 controllers can be mounted in several different configurations. Three RA60, RA80, or RA81 units can be installed in a single, low-height, standard-width Digital cabinet or a single disk drive can be mounted in a cabinet together with the TU80 magnetic tape unit. The RA80 and RA81 disk drives include Winchester technology for reliable fixed-data storage, and the RA60 disk drive provides removable storage media. Up to four disk drives in any combination can operate with either an HSC50 or UDA50 controller. The RA60, RA80, and RA81 disk drives contain dual ports and an intelligent controller can be connected to each port. When a second UDA50 controller is included, the disk information can be shared between the controllers. The simultaneous access to the data in a drive through the dual ports, however, is not supported by Digital's operating systems or diagnostic software.

RA60 Removable-media Disk Drive

The RA60 is a medium-capacity disk drive that stores up to 205 Mbytes of formatted data on a removable disk cartridge. The RA60 drive is available mounted in a cabinet or as a separate unit to be mounted in a standard cabinet 48.26 centimeters (19.0 inches) wide. The unit requires a vertical mounting space of 26.7 centimeters (10.5 inches) and a cabinet depth of 91.4 centimeters (36.0 inches). When installed with slide mounts, up to three disk drives can be installed in a Digital cabinet. The media is loaded and removed from the top of the unit. The RA60 incorporates new recording methods, microprocessor-controlled diagnostics, enhanced servo technology, 170-bit error correcting code and modular design. It also contains dual-access ports, which allow operation with two intelligent controllers. Figure 12-3 shows the RA60 unit mounted at the top of a cabinet.

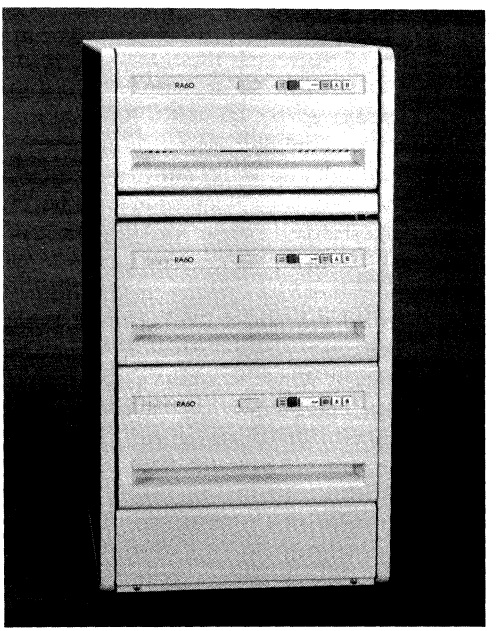


Figure 12-3 • RA60 Removable-media Disk Drive

The *performance specifications* of the RA60 disk drive are
▪ Formatted capacity: 205 Mbytes.
▪ Peak transfer rate: 198 Mbytes per second.
▪ Access time: 50 milliseconds (average).
▪ Seek time: 41.7 milliseconds (average).
▪ Track-to-track seek time: 6.7 milliseconds.
The *media specifications* of the RA60 disk drive are
▪ Rotational speed: 3,600 revolutions per minute.
▪ Disk surfaces: 6 data and 4 protective.
▪ Tracks per surface: 1,600.
▪ Sectors per track: 43 (16-bit words).
▪ Bytes per sector: 512.

RA80 Fixed-media Disk Drive

The RA80 is a medium-capacity disk drive that incorporates Winchester drive technology and provides up to 121 Mbytes of formatted fixed-data storage. The RA80 drive is available mounted in a cabinet or as a separate unit to be mounted in a standard cabinet 48.26 centimeters (19.0 inches) wide. The unit requires a vertical mounting space of 26.7 centimeters (10.5 inches) and a cabinet depth of 91.4 centimeters (36.0 inches). Up to three RA80 units can be installed in a cabinet 106.7 centimeters (42 inches) high. The RA80 incorporates a rotary positioner motor, computer-designed positioner arms, and lightweight Winchester head suspension. It also contains dual-access ports to allow operation with two intelligent controllers. Figure 12-4 shows two RA80 units mounted in a cabinet.

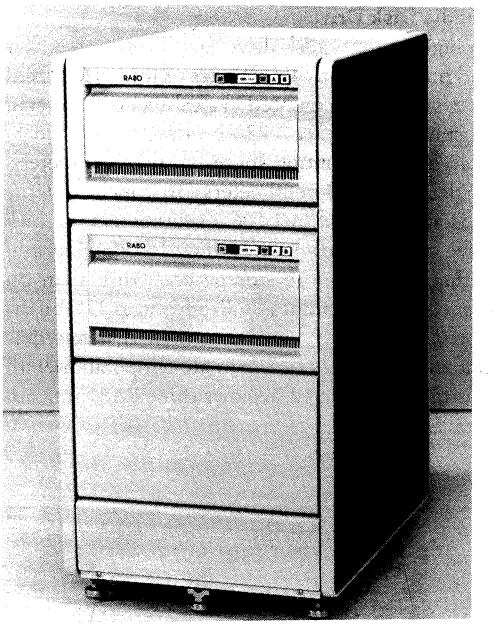


Figure 12-4 • RA80 Fixed-media Disk Drive

The *performance specifications* for the RA80 disk drive are

- Formatted capacity: 121 Mbytes.
- Peak transfer rate: 1.2 Mbytes per second.
- Access time: 33.3 milliseconds (average).
- Seek time: 25 milliseconds (average).
- Track-to-track seek time: 6 milliseconds.
- Latency time: 8.33 milliseconds (average).

The *media specifications* are

- Rotational speed: 3,600 revolutions per minute.
- Disk surfaces: 7 data and 1 servo.
- Tracks per surface: 1,092.
- Sectors per track: 31 (16-bit words).
- Bytes per sector: 512.

RA81 Fixed-media Disk Drive

The RA81 is a high-capacity disk drive that incorporates Winchester drive technology and provides up to 456 Mbytes of formatted fixed-data storage. The RA81 drive is available mounted in a cabinet or as a separate unit to be mounted in a standard cabinet 48.26 centimeter (19.0 inches) wide. The unit requires a vertical mounting space of 26.7 centimeters (10.5 inches) and a cabinet depth of 91.4 centimeters (36.0 inches). Up to three RA81 units can be installed in a cabinet 106.7 centimeters (42 inches) high. The RA81 disk drive incorporates an encoding/decoding scheme in the read-write system that permits more storage on a disk than on drives using conventional encoding techniques. It also features 170-bit error correction codes and more than 17,000 spare sectors for dynamic defect relocation. The RA81 contains dual-access ports to allow operation with two separate intelligent controllers. Figure 12-5 shows three RA81 units mounted in the standard Digital cabinet.

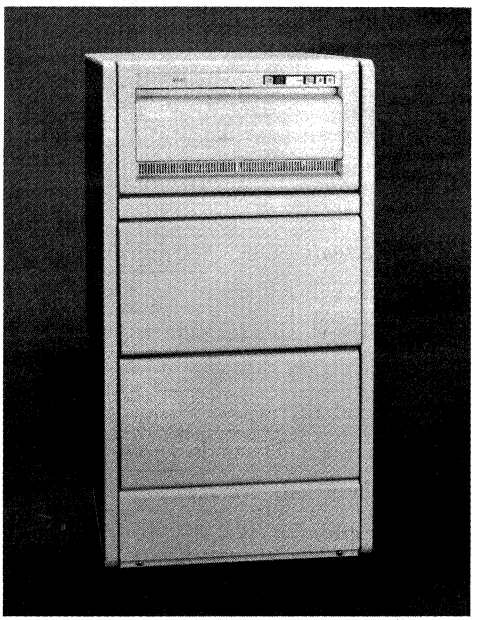


Figure 12-5 ▪ RA81 Fixed-media Disk Drive

The *performance specifications* for the RA81 disk drive are

- Formatted capacity: 456 Mbytes.
- Peak transfer rate: 2.2 Mbytes per second.
- Access time: 36.3 milliseconds (average).
- Seek time: 28 milliseconds (average).
- Track-to-track seek time: 6 milliseconds.
- Latency time: 8.33 milliseconds (average).

The *media specifications* are

- Rotational speed: 3,600 revolutions per minute.
- Disk surfaces: 7 data and 1 servo.
- Tracks per surface: 2,496.
- Sectors per track: 51.
- Bytes per sector: 512.

UNIBUS Disk-drive Subsystems

The RL02 and RC25 are low-cost disk drives that operate with the UNIBUS of the VAX processors and can be conveniently installed into a standard Digital low-height cabinet. The RC25 disk drive is also available as a table-top unit. The units are attached to the processor through a UNIBUS controller and up to four RL02 drives and two RC25 drives can operate with each controller. The RL02 disk drive is not supported as a system disk but is used as a data file residency device for transferring data to and from other Digital systems.

RL02 Cartridge Disk Drive

The RL02 is a low-cost disk drive that stores 10.4 Mbytes of formatted data on a removable-disk cartridge. The RL02 drive is available mounted in a cabinet or as a separate unit to be mounted in a standard cabinet 48.26 centimeters (19.0 inches) wide. The unit requires a vertical mounting space of 26.7 centimeters (10.5 inches) and a cabinet depth of 76.2 centimeters (30 inches). The disk cartridge is loaded and removed from the top of the unit. The RA60 incorporates an imbedded closed-loop servopositioning system that improves the integrity of the data by continually sampling the servo information with the same head that is used to read and write the data. A cyclic redundancy check is also performed on the data transfers between the drive and controller. Reliable reading and recording is performed by a phased-locked clock and frequency modulation techniques. Data is transferred to the processor UNIBUS though sequential block direct memory access (DMA) transfers. Figure 12-6 shows the RL02 unit and removable-disk cartridge.

Figure 12-6 ▪ RL02 Removable-Media Disk Drive

The *performance specifications* for the RL02 disk drive are

- Formatted capacity: 10.4 Mbytes.
- Peak transfer rate: 512 Kbytes per second.
- Access time: 67.5 milliseconds (average).
- Seek time: 55 milliseconds (average).
- Track-to-track seek time: 15 milliseconds.

The *media specifications* are

- Rotational speed: 2,400 revolutions per minute.
- Disk surfaces: 2 data.
- Tracks per surface: 512.
- Sectors per track: 40.
- Bytes per sector: 256.

RC25 Fixed/Removable Disk Drive

The RC25 is a small and compact disk subsystem that provides 52 Mbytes of formatted data storage. It operates with the UNIBUS on VAX processor systems and contains 26 Mbytes of storage on fixed-disk media and 26 Mbytes on removable-disk media. The RC25 is available as a tabletop unit or as a unit that can be mounted in a standard cabinet or rack 48.26 centimeters (19 inches) wide. The tabletop unit is 25.4 centimeters (10 inches) wide, 25.4 centimeters (10 inches) high, and 50.8 centimeters (20 inches) deep. The RC25 incorporates Winchester technology for the fixed-disk media and an intelligent controller and microdiagnostics for maintenance functions. The RC25 is available with or without a controller. A maximum of two drives can be operated with each controller. Reliable operation and error-free transmission is ensured by 170-bit error detection and correction code, by automatic retry and vectoring, by an embedded servo control, and by the replacement of bad data blocks. The RC25 is compatible with Digital Storage Architecture by using the mass storage control protocol. The RC25 tabletop unit is shown in figure 12-7.

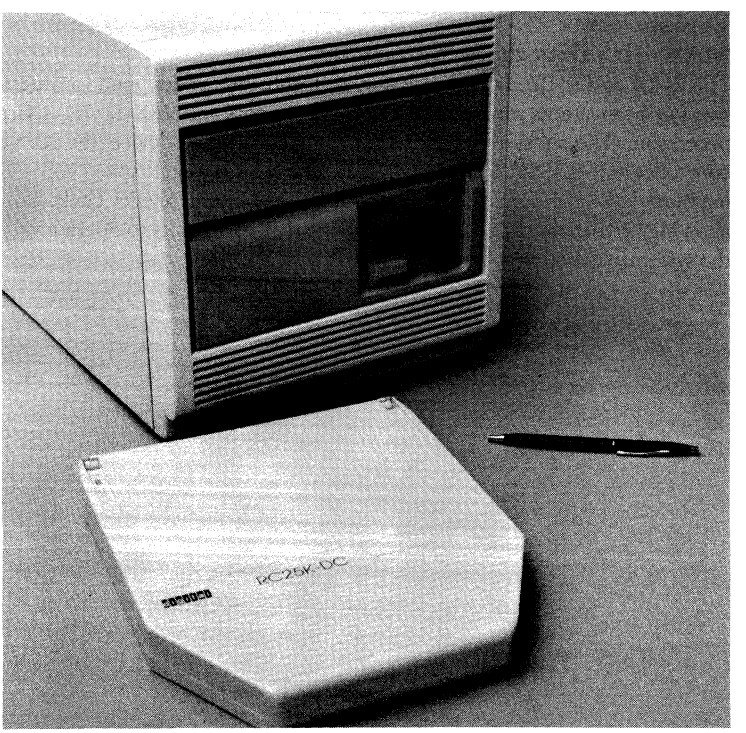


Figure 12-7 • RC25 Fixed/Removable Disk Drive

The *performance specifications* of the RC25 disk drive are

- Formatted capacity: 56 Mbytes total.
- Peak transfer rate: 1.25 Mbytes per second.
- Access time: 45.5 milliseconds (average).
- Seek time: 10 milliseconds (average).
- Track-to-track seek time: 35 milliseconds (average).
- Rotational latency: 10.5 milliseconds (average).
- Rotational speed: 3,633 revolutions per minute.

The *media specifications* of the RC25 disk drive are

- Disk surfaces: 16 data and 1 servo.
- Tracks per surface: 1,260 data and 4 diagnostic.
- Sectors per track: 50 (16-bit format).
- Bytes per sector: 512 (16-bit format).

MASSBUS Disk-drive Subsystems

The RM05 and RP07 are high-capacity disk subsystems designed for intensive I/O applications run on VAX-11/750 processors and VAX-11/780 series processors. The subsystems consist of the mass storage device, disk-drive controller and MASSBUS adapter (MBA). The MBA is a high-speed controller module that mounts in the computer's bus-interconnects backplane, allowing up to eight disk drives or magnetic tape formatters to operate with the system.

RM05 Removable-media Disk Drive

The RM05 is a removable-media disk storage subsystem that operates with the MASSBUS on VAX-11/750 processors and VAX-11/780 series processors. It is designed for intensive data-transfer applications between the host processor and the disk drive, providing 256 Mbytes of formatted storage on the removable disk. The RM05 is contained in two cabinets 92.0 centimeters (36.25 inches) high, one containing the drive and controller and one utility cabinet that contains the RM05 adapter. Space is included in the utility cabinet to mount an additional adapter. Up to eight disk-drive units can be operated with each controller. The storage media is loaded from the top of the drive cabinet. High-level data throughput is achieved by overlapping the seek operations with the simultaneous transfer of control information and data on the MASSBUS. That permits mid-transfer seek operations that automatically address the next data block on the same or next higher cylinder and allows data to be transferred in blocks. Data is transferred from sequential blocks on the device to memory using direct memory access (DMA) operations. The RM05 disk subsystem includes dual ports for operation with two separate controllers; however, the dual-port capability is not supported by Digital. The subsystems can be statically shared by two processors or connected to one processor through two controllers. The disk subsystem and removable disk media are shown in figure 12-8.

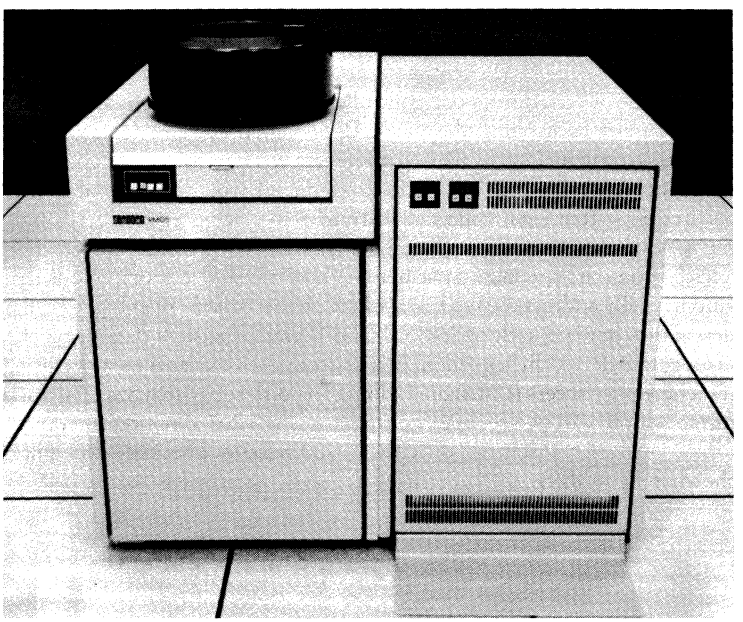


Figure 12-8 ▪ RM05 Removable-media Disk Drive

The *performance specifications* for the RM05 disk drive are

- Formatted capacity: 256 Mbytes.
- Peak transfer rate: 1,200 Kbytes per second.
- Access time: 38.3 milliseconds (average).
- Seek time: 30 milliseconds (average).
- Track-to-track seek time: 6 milliseconds.
- Latency time: 8.3 milliseconds (average).

The *media specifications* for the RM05 disk drive are

- Rotational speed: 3,600 revolutions per minute.
- Disk surfaces: 19 data and 1 servo.
- Tracks per surface: 823.
- Sectors per track: 32 (16-bit format).
- Bytes per sector: 512 (16-bit format).

RP07 Fixed-media Disk Drive

The RP07 is a high-capacity disk-drive subsystem that operates with the MASSBUS on the VAX-11/750 processor and VAX-11/780 series processors. It incorporates Winchester drive technology and provides up to 516 M-bytes of formatted fixed data storage. The RP07 drive and controller are contained in a cabinet 117 centimeters (46.1 inches) high. Up to eight drive units can be operated from one controller. The RP07 unit includes a microprocessor for controlling the major drive functions, and fault-isolation diagnostic programs to facilitate maintenance. The disk media and servo heads are contained in a sealed, contaminant-free disk assembly that ensures error-free data. High data throughput is achieved by overlapping the seek operations with the simultaneous transfer of control information and data on the MASSBUS. That permits mid-transfer seek operations that automatically address the next data block on the same or next higher cylinder, allows implied seek operations, and permits transfers data in blocks. Data is transferred in sequential blocks from the device to memory using direct memory access (DMA) operations. The disk subsystem has dual ports for operation with two separate controllers. The RP07-D option is required for a 2.2-Mbyte transfer rate on VAX-11/780 systems. The RP07 disk unit is shown in figure 12-9.

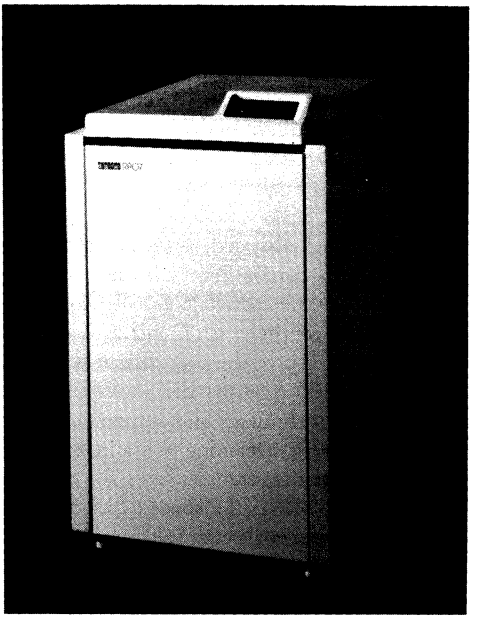


Figure 12-9 • RP07 Fixed-media Disk Unit

The *performance specifications* for the RP07 disk drive are

- Formatted capacity: 516 Mbytes.
- Peak transfer rate: 1.3 Mbytes per second (2.2 Mbytes per second optional).
- Access time: 31.3 milliseconds (average).
- Seek time: 28 milliseconds (average).
- Track-to-track seek time: 5 milliseconds.
- Latency time: 8.33 milliseconds (average).

The *media specifications* for the RP07 disk drive are

- Rotational speed: 3,633 revolutions per minute.
- Disk surfaces: 16 data and 1 servo.
- Tracks per surface: 1,260 data and 4 diagnostic.
- Sectors per track: 50 (16-bit format).
- Bytes per sector: 512 (16-bit format).

▪ Magnetic Tape Subsystems

Digital provides a complete series of industry-compatible magnetic tape subsystems to operate with the UNIBUS and MASSBUS on VAX systems and with the HSC50 intelligent controller in a VAXcluster system. The magnetic tape units provide reliable medium- and high-density backup storage for the Digital disk drives.

TU80 Magnetic Tape Unit

The TU80 is a single-density, dual-speed, 9-track tape subsystem that operates with the UNIBUS and is compatible with phase encoding (PE) ANSI industry standards. The TU80 tape drive, controller, and power system are contained in a cabinet 106.7 centimeters (42 inches) high. The tape drive is horizontally mounted at the top of the unit. Space is available within the cabinet for mounting an RA80 or RA81 disk drive subsystem. The TU80 incorporates streaming tape technology and automatically selects speeds of 25 or 100 inches per second for streaming operations and up to 25 inches per second, depending on the data transfer rates, for start/stop operations. The controller automatically selects the speeds, based on the type of data transfer. The tape subsystem is connected to the VAX processor through a shielded cable and a UNIBUS adapter module. The TU80 conforms to the ANSI standard for phase encoding (PE) with 1,600 bits per inch on ½-inch, 9-track tape. The TU81 uses the software protocol of the Digital Storage Architecture to prefetch commands and data for high-speed operation. The TU80 unit is shown in figure 12-10.

Figure 12-10 ▪ TU80 Magnetic Tape Unit

The *performance* and *media specifications* for the TU80 magnetic tape unit are

- Read/write speed: 25 and 100 inches per second (streaming) and 25 inches per second (maximum start/stop).
- Data transfer speed: 160 Kbytes per second maximum.
- Rewind speed: 192 inches per second.
- Rewind time: 2.5 minutes per 2,400-foot (731.5-meter) reel.
- Tape tracks: 9.
- Recording method: PE conforms to ANSI standard X3.39-1973.
- Recording density: 1,600 bits per inch (PE).
- Capacity: 40 Mbytes (PE).
- Magnetic tape: ½-inch, 9-track, conforms to ANSI standard X3.40-1981.

TU81 Magnetic Tape Unit

The TU81 is a dual-density, dual-speed, 9-track tape subsystem that operates with the UNIBUS and is compatible with the ANSI standard for group code recording (GCR) and phase encoding (PE). The TU81 tape drive, controller, and power system are contained in a cabinet 106.7 centimeters (42 inches) high. The tape drive is mounted horizontally at the top of the unit and space is available within the cabinet for mounting an RA80 or RA81 disk-drive subsystem. The TU81 incorporates streaming tape technology and automatically selects speeds of 25 and 75 inches per second for streaming operations, and up to 25 inches per second for start/stop operations, depending on the data transfer rates. The controller automatically selects the speed that is appropriate for the type of data transfer. The tape subsystem connects to the VAX processor through a shielded cable and a UNIBUS adapter module. As a dual-density drive, the TU81 conforms to the ANSI standard for group code recording (GCR) with 6,200 bits per inch and for phase encoding (PE) with 1,600 bits per inch on ½-inch, 9-track tape. The TU81 used the software protocol of the Digital Storage Architecture, which enables high-speed operation by the prefetching of commands and data. The TU81 unit is shown in figure 12-11.

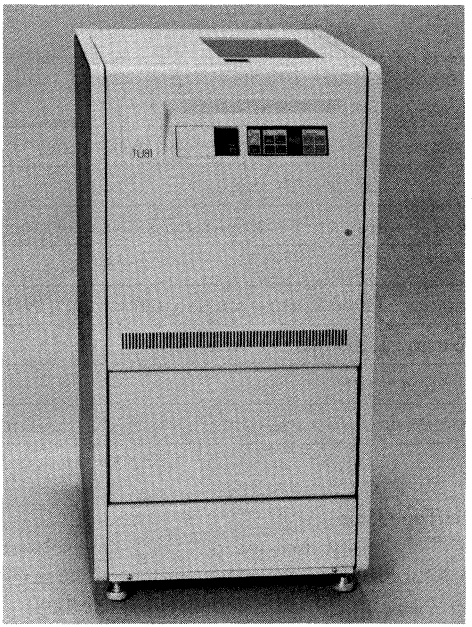


Figure 12-11 ▪ TU81 Magnetic Tape Unit

The *performance* and *media specifications* for the TU81 magnetic tape unit are as follows:

- Read/write speed: 25 and 75 inches per second (maximum streaming) and 25 inches per second (maximum start/stop).
- Data transfer speed: 468 Kbytes per second.
- Rewind speed: 192 inches per second.
- Rewind time: 2.5 minutes per 2,400-foot (731.5-meter) reel.
- Tape tracks: 9.
- Recording method: GCR conforms to ANSI standard X3.54-1976 and PE conforms to ANSI standard X3.39-1973.
- Recording density: 6,250 bits per inch (GCR) and 1,600 bits per inch (PE).
- Storage capacity: 140 Mbytes (GCR) and 40 Mbytes (PE).
- Magnetic tape: 1/2-inch, 9-track, conforms to ANSI X3.40-1981.

TA78 Magnetic Tape Unit

The TA78 is a dual-density, 9-track tape subsystem that operates as a MASSBUS storage device with the VAX-11/750 processor, VAX-11/780 series processor, or with the HSC50 intelligent controller in a VAXcluster configuration. The TA78 transport conforms to the IBM and ANSI industry-standard recording density of 6,250 bits per inch for group code recording (GCR) and 1,600 bits per inch for phase encoding (PE). The recording density is selected by program. The TA78 is contained in a cabinet 153 centimeters (60.2 inches) high and contains a vertically mounted tape drive, a tape formatter and a power system. The tape is automatically threaded through the transport. Up to three separately housed TU78 transports can be operated with one tape formatter. The TA78 performs self-test diagnostics during startup and when data is not being transferred to the drive. A video terminal can be connected to the unit to facilitate the diagnostic and fault isolation. The tape subsystem connects to the VAX processor through a shielded cable and a MASSBUS adapter module. The TA78 contains dual ports to allow the unit to be controlled by two HSC50 server units. The TA78 magnetic tape unit is shown in figure 12-12.

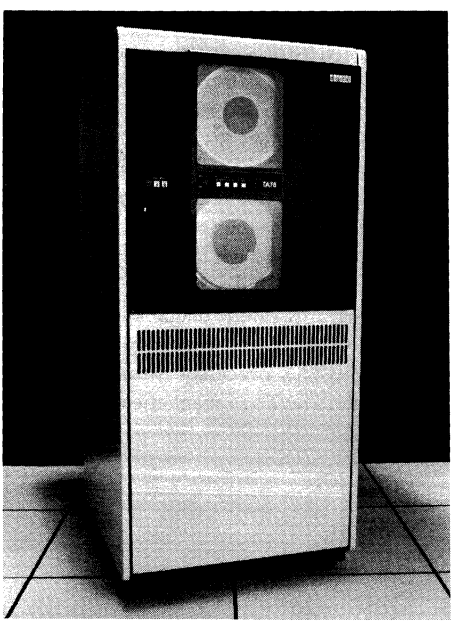


Figure 12-12 • TA78 Magnetic Tape Unit

The *performance* and *media specifications* for the TA78 magnetic subsystem are as follows:

- Read/write speed: 125 inches per second.
- Data transfer rate: 200 Kbytes per second (PE) and 781 Kbytes per second (GCR).
- Rewind speed: 440 inches per second.
- Rewind time: 65 seconds per 2,400-foot (731.5-meter) reel.
- Tape tracks: 9.
- Recording method: GCR conforms to ANSI standard X3.54-1976 and PE conforms to ANSI standard X3.39-1973.
- Recording density: 6,250 bits per inch (GCR) and 1,600 bits per inch (PE).
- Storage capacity per 2,400-foot reel: 140 Mbytes at 6,250 bits per inch and 40 Mbytes at 1,600 bits per inch.
- Magnetic tape: 1/2-inch conforms to ANSI standard X3.40-1981.

TU77 Magnetic Tape Drive

The TU77 is a dual-density, 9-track tape subsystem that operates with the MASSBUS on a VAX-11/750 processor or VAX-11/780 series processor. The TU77 transport conforms to the industry standard recording density of 1,600 bits per inch phase encoding (PE) and 800 bits per inch nonreturn to zero (NRZ). The recording density is selectable by the program. The TU77 is contained in a cabinet 153 centimeters (60.2 inches) high and provides a vertically mounted tape drive, a tape formatter, and a power system. The tape is automatically threaded through the transport. Up to four separately housed transports can be operated with one TU77 tape formatter. The TU77 contains self-test features, online diagnostics, and fault isolation diagnostics to facilitate maintenance and ensure reliable operation. The tape subsystem connects to the VAX processor through a shielded cable and a MASSBUS adapter module. The TU77 magnetic tape unit is shown in figure 12-13.

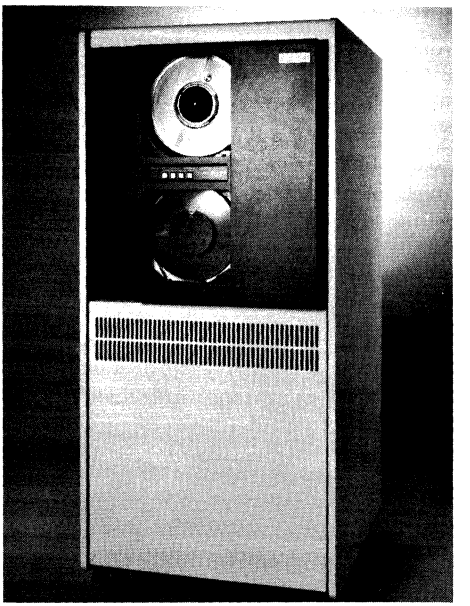


Figure 12-13 • TU77 Magnetic Tape Unit

The *performance* and *media specifications* for the TU77 magnetic tape unit are as follows:

- Read/write speed: 125 inches per second.
- Data transfer speed: 200 Kbytes per second.
- Rewind speed: 440 inches per second.
- Rewind time: 70 seconds per 2,400-foot (731.5-meter) reel.
- Tape tracks: 9.
- Recording method: PE conforms to IBM standards and ANSI standard X3.39-1973.
- Recording density: 1,600 bits per inch (PE) or 800 bits per inch (NRZ).
- Storage capacity per 2,400-foot reel: 40 Mbytes with 8-Kbyte blocks at 1,600 bits per inch and 20 Mbytes with 8-Kbyte blocks at 800 bits per inch.
- Magnetic tape: ½-inch, conforms to ANSI standard X3.40-1981.

TU78 Magnetic Tape Unit

The TU78 is a dual-density, 9-track tape subsystem that operates with the MASSBUS on a VAX-11/750 processor or VAX-11/780 series processor and with the HSC50 intelligent controller in a VAXcluster configuration. The TU78 transport conforms to the IBM and ANSI industry-standard recording density of 6,250 bits per inch for group code recording (GCR) and 1,600 bits per inch for phase encoding (PE). The TU78 is contained in a cabinet 153 centimeters (60.2 inches) high and contains a vertically mounted tape drive, a tape formatter, and a power system. The tape is automatically threaded through the transport. Up to four separately housed transports can be operated with one tape formatter. The TU78 contains self-test features, online diagnostics, and fault isolation diagnostics to facilitate maintenance and ensure reliable operation. The tape subsystem connects to the VAX processor through a shielded cable and a MASSBUS adapter module. The TU78 provides dual port capabilities as an option to enable the drive to operate with two controllers. The TU78 magnetic tape unit is shown in figure 12-14.

Figure 12-14 ▪ TU78 Magnetic Tape Unit

The *performance* and *media specifications* for the TU78 magnetic tape unit are as follows:

- Read/write speed: 125 inches per second.
- Data transfer rate: 781 Kbytes per second.
- Rewind speed: 440 inches per second.
- Rewind time: 70 seconds per 2,400-foot (731.5-meter) reel.
- Tape tracks: 9.
- Recording method: GCR at 6,250 bits per inch and PE at 1,600 bits per inch.
- Recording density: 6,250 and 1,600 bits per inch.
- Storage capacity per 2,400-foot (731.5-meter) reel: 145 Mbytes at 6,250 bits per inch.
- Magnetic tape: 1/2-inch conforms to ANSI standard X3.40-1981.

TS05 Magnetic Tape Unit

The TS05 is a small, dual-speed magnetic tape subsystem that conforms to the ANSI industry standard for 1,600-bits-per-inch phase encoding (PE). The device connects to the UNIBUS of the VAX processors through an interface and cable. The TS05 magnetic tape unit occupies a vertical mounting space of 22 centimeters (8.7 inches) in a Digital cabinet or standard cabinet or rack 48.26 centimeters (19 inches) wide. The transport speed can be preset to 25 inches per second or 100 inches per second. The subsystem consists of the tape transport drive and tape formatter and an interface/controller module that installs in the UNIBUS backplane slot. The TS05 is a high-performance subsystem that incorporates streaming tape technology and automatically threads the 1/2-inch tape in the transport. The TS05 magnetic tape unit is shown in figure 12-15.

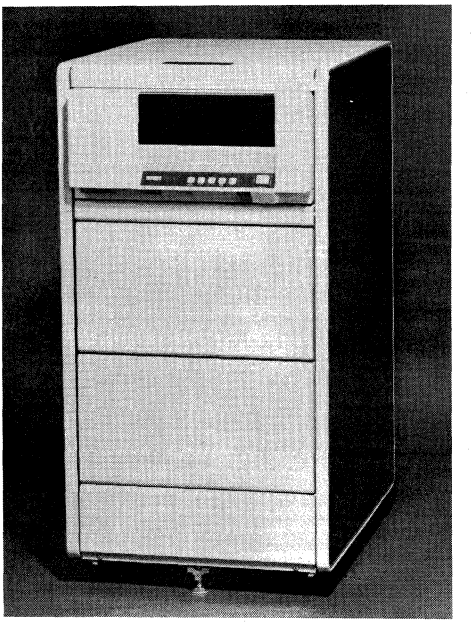


Figure 12-15 ▪ TS05 Magnetic Tape Unit

The *performance* and *media specifications* for TS05 magnetic tape subsystem are as follows:

- Read/write speed: 25 or 100 inches per second (preset by user).
- Data transfer rate: 40 Kbytes or 160 Kbytes per second.
- Rewind speed: 180 inches per second (maximum).
- Rewind time: 2.8 minutes per 2,400-foot (731.5-meter) reel.
- Recording method: PE conforms to ANSI standards.
- Recording density: 1,600 bits per inch.
- Storage capacity per 2,400-foot reel: 40 Mbytes using 8-Kbyte blocks.
- Magnetic tape: 1/2-inch.

· VAX-11/725 Processor

Processor Type	
Microcontrol store instruction time	270 ns
Control store size	16-KB (24-bit word) ROM
Internal data path	32 bits
Instruction buffer size	4-byte lookahead

VAX Instruction Set	
Number of 32-bit registers	16
Number of basic operations	304
Optional instructions	56 with FP730 floating-point accelerator
Number of priority interrupt levels	32
Number of addressing modes	9
PDP-11 compatibility	
Data types	Integer, floating-point, packed decimal, character string, variable-bit fields, numeric strings, and queues

Main Memory	
Virtual address space	4 GB
Physical address space	16 MB (total)
Address lines	24 bits
Parity	7-bit error-correcting code (ECC) per 32-bit longword
Memory cycle time	810 ns per 32-bit read or write

CPU Address Translation Buffer	
Size	128 address translations

CPU Clocks	
Realtime clock	Programmable, crystal controlled, 0.01% accuracy, 1 μs resolution
Time-of-year clock	

Console Subsystem	
Microprocesser	8085A
Load device	Two TU58 cartridge tape drives
Capacity per cartridge	256 KB (512 records of 512 bytes each)
Bit density	800 b/in
Tape length	140 ft
Transfer rate	19.2 KB/s
Access time	9.3 s (avg) 28 s (max.)
Tape speed	30 in/s read/write 60 in/s search

RC25 Disk Subsystem	
Disk type	8-in Winchester
Formatted capacity	26 MB fixed disk, 26 MB removable disk
Performance	35 ms seek time (20 ms in long request queues)
Access time including latency	45.5 ms (avg)
Data transfer rate	1,250 KB/s peak

UNIBUS Adapter	
I/O transfer rate	1.5 MB/s (max.)
Interrupts	Directly vectored

CPU Options	
Memory expansion	1 MB increments (64-KB chip)
Communication interfaces	8-, 16-, and 24-line asynchronous interfaces, multifunction controller, and Ethernet controller
FP730 floating-point accelerator	Single- and double-precision, G and H floating-point, and POLY, EMOD, and MULL instructions

Operating Environment	
Temperature	10°C to 40°C (50°F to 104°F)
Relative humidity	10% to 90% (noncondensing)
Altitude	2,400 m (8,000 ft) max.
Heat dissipation (estimated)	492 kcal/h (1952 Btu/h) max.
Acoustics level (estimated)	55 dB

Processor Power Requirements	
ac line voltage tolerance	90 to 128 V, 60 Hz 180 to 256 V, 50 Hz
Line frequency tolerance	47 to 63 Hz
Phases	1
Surge current	17 A at 93 Vrms 8.5 A at 180 Vrms
Steady-state current	7.1 A at 93 Vrms 3.6 A at 180 Vrms
ac power consumption	575 W (max.)

Physical Characteristics	
Weight	93 kg (205 lb)
Height	62.2 cm (24.5 in)
Width	44.5 cm (17.5 in)
Depth	72.4 cm (28.5 in)

Typical Power and Thermal Dissipation

Option	*Watts*	*Heat Dissipation*
FP730 floating-point accelerator	51	43 kcal/h (172 Btu/h)
DMF32 communication controller	55	47 kcal/h (188 Btu/h)
DEUNA Ethernet controller	70	60 kcal/h (239 Btu/h)

▪ VAX-11/730 Processor

Processor Type	
Microcontrol store instruction time	270 ns
Control store size	16-KB (24-bit word) ROM 17 Kwords with IDC
Internal data path	32 bits
Instruction buffer size	4-byte lookahead

VAX Instruction Set	
Number of 32-bit registers	16
Number of basic operations	304
Optional instructions	90 instructions accelerated with FP730 floating-point accelerator option
Number of priority interrupt levels	32
Number of addressing modes	9
PDP-11 compatibility	
Data types	Integer, floating-point, packed decimal, character string, variable-bit fields, numeric strings, and queues

Main Memory	
Virtual address space	4 GB
Physical address space	3 MB total
Address lines	24 bits
Parity	7-bit error-correcting code (ECC) per 32-bit longword
Memory cycle time	810 ns per 32-bit read or write

CPU Address Translation Buffer	
Size	128 address translations

CPU Clocks	
Realtime clock	Programmable, crystal controlled, 0.01% accuracy, 1 μs resolution
Time-of-year clock	

Console Subsystem	
Interface	8085A microprocesser-controlled
Console device	Two TU58 cartridge tape drives
Capacity per cartridge	256 KB (512 records of 512 bytes each)
Bit density	800 b/in
Tape length	140 ft
Transfer rate	19.2 KB/s
Access time	9.3 s (avg) 28 s (max.)
Tape speed	30 in/s read/write 60 in/s search

UNIBUS Adapter	
I/O transfer rate	1.5 MB/s (max.)
Interrupts	Directly vectored

CPU Options	
Memory expansion	1 MB increments (64-KB chip)
Communication interfaces	8-, 16-, and 24-line asynchronous interfaces, multifunction controller, and Ethernet controller
H7750 memory battery backup	One per memory controller
FP730 floating-point accelerator	Single- and double-precision, G and H floating-point, and POLY, EMOD, and MULL instructions
TU80 magnetic tape controller	One per system
RUC25 disk controller	One per system

R80/RL02 System	
R80 disk drive	Fixed media
Formatted capacity	121 MB
Peak transfer rate	1.2 MB/s
Average access time	33.3 ms
Average seek time	25 ms
Average latency time	8.33 ms
RL02 disk drive	Removable media
Formatted capacity	10.4 MB
Peak transfer rate	512 KB/s
Average access time	67.5 ms
Average seek time	55 ms
Average latency time	12.5 ms

R80/TSU05 System	
R80 disk drive	Refer to R80/RL02 system
TSU05 magnetic tape drive	1/2-in removable media
Recording density	1,600 b/in
Read/write speed	25/100 in/s
Capacity per 2,400 ft reel	40 Mbytes with 8K blocks at 1,600 b/in
Data transfer speed	40/160 b/s (max.)
Rewind speed	180 in/s (max.)
Rewind time	2.8 min. per 2,400-ft reel

Operating Environment	
Temperature	
R80/RL02 system	10°C to 40°C (50°F to 104°F)
R80/TSU05 system	15°C to 32°C (59°F to 90°F)
Relative humidity	
R80/RL02 system	10% to 90% (noncondensing)
R80/TSU05 system	20% to 80% (noncondensing)
Altitude	2,400 m (8,000 ft) max.

Processor Power Requirements	
ac line voltage	90 to 128 V, 60 Hz 180 to 256 V, 50 Hz
Line frequency	47 to 63 Hz
Phases	1
R80/RL02 System	
Steady-state current	15 A at 90 Vrms (120 V) 7 A at 180 Vrms (240 V)
Surge current	32 A at 90 Vrms (120 V) 12 A at 180 Vrms (240 V)
Heat dissipation	1,037 kcal/h (4,109 Btu/h) (max.)
ac power consumption	1,205 W
R80/TSU05 System	
Steady-state current	17 A at 90 Vrms (120 V) 9 A at 180 Vrms (240 V)
Surge current	34 A at 90 Vrms (120 V) 14 A at 180 Vrms (240 V)
Heat dissipation	1,119 kcal/hr (4,433 Btu/hr) (maximum)
ac power consumption	1,300 W

Physical Characteristics	
Weight	
CPU and cabinet only	45.4 kg (100 lb)
R80/RL02 system	249.7 kg (550 lb)
R80/TSU05 system	290 kg (639 lb)
Height	106.2 cm (41.8 in)
Width	73.7 cm (29 in)
Depth	76.2 cm (30 in)

Typical Power and Thermal Dissipation		
Option	*Watts*	*Heat Dissipation*
Fully-configured general-purpose expansion cabinet (H9642-F)	1,224	1,054 kcal/h (4,173 Btu/h)
FP730 floating-point accelerator	51	43 kcal/h (172 Btu/h)
DMF32 communication controller	55	47 kcal/h (188 Btu/h)
DEUNA Ethernet controller	70	60 kcal/h (239 Btu/h)
UDA50 disk controller	90	77.5 kcal/h (307 Btu/h)
H7750 memory battery backup	25	22 kcal/h (85 Btu/h)

▪ VAX-11/750 Processor

Processor Type	
Microcontrol store instruction time	320 ns
Control store size	6 Kwords (80-bit words) ROM
Internal data path	32 bits
Instruction buffer size	8-byte lookahead
Memory cache	4-KB direct-mapped
System data transfer rate	5 MB/s (max.)

VAX Instruction Set	
Number of 32-bit registers	16
Number of basic operations	304
Number of priority interrupt levels	32
Number of addressing modes	9
Data types	Integer, floating-point, packed decimal, character string, variable-bit fields, numeric strings, and queues

Main Memory	
Virtual address space	4 GB
Physical address space	8 MB total
Parity	7-bit error-correcting code (ECC) per 32-bit quadword
Memory cycle time	800 ns per 32-bit read, 640 ns per 32-bit write, 400 ns (effective) with cache

CPU Address Translation Buffer	
Size	512 address translations
Typical hit ratio	98% to 99%

CPU Clocks	
Realtime clock	Programmable, crystal controlled, 0.01% accuracy, 1 μs resolution
Time-of-year clock	Includes battery backup for more than 100 hours

Console Subsystem	
Interface	Integral (UIM module)
Console device	TU58 cartridge tape drive
Capacity per cartridge	256 KB (512 records of 512 bytes each)
Bit density	800 b/in
Tape length	140 ft
Transfer rate	19.2 KB/s
Access time	9.3 s (avg) 28 s (max.)
Tape speed	30 in/s read/write 60 in/s search

CPU Options	
Memory expansion	1-MB increments (64-KB chip)
Communication interfaces	8-, 16-, and 24-line asynchronous interfaces, multifunction controller, and Ethernet controller
DR750 general purpose interface	One per system; 32-bit parallel data interface
DW750 UNIBUS adapter	One per system
H7112 memory battery backup	One per memory controller
FP750 floating-point accelerator	Single- and double-precision, F and D floating-point and POLY, EMOD, and MULL instructions
KU750 floating-point option	G and H extended-range floating-point data types
CI750 adapter	Computer interconnect for VAXcluster

Operating Environment	
Temperature	10°C to 40°C (50°F to 104°F)
Relative humidity	10% to 90% (noncondensing)
Altitude	2,400 m (8,000 ft) max.
Heat dissipation	1,460 kcal/h (5,800 Btu/h) (max.)

Processor Power Requirements	
ac line voltage tolerance	104 to 128 V, 60 Hz 191 to 233 V, 50 Hz 208 to 254 V, 50 Hz
Line frequency tolerance	47 to 63 Hz
Phases	Single
Steady-state current	30 A at 90 Vrms (120 V) 15 A at 180 Vrms (240 V)
Surge current	100 A
ac power consumption	1700 W (max.)

Physical Characteristics	
Weight	182 kg (400 lb)
Height	106.2 cm (41.8 in)
Width	73.7 cm (29 in)
Depth	76.2 cm (30 in)

Typical Power and Thermal Dissipation		
Option	*Watts*	*Heat Dissipation*
Fully configured CPU cabinet	1,700	1,460 kcal/h (5,797 Btu/h)
Fully configured general-purpose expansion cabinet (H9642-F)	1,224	1,054 kcal/h (4,173 Btu/h)
RH750 MASSBUS adapter	77	67 kcal/h (262 Btu/hr)
DW750 UNIBUS adapter	77	67 kcal/hr (262 Btu/h)
CI750 computer interconnect	56	49 kcal/h (191 Btu/h)
DR750 general purpose interface	100	86 kcal/h (341 Btu/h)
FP750 floating-point accelerator	90	78 kcal/h (307 Btu/h)
KU750-A user writable control store	18	16 kcal/h (62 Btu/h)
DMF32 communication controller	55	47 kcal/h (188 Btu/h)
DEUNA Ethernet controller	70	60 kcal/h (239 Btu/h)
H7112 memory battery backup	25	22 kcal/h (85 Btu/h)

▪ VAX-11/780 Processor

Processor Type	
Microcontrol store instruction time	200
Control store size	6-KB (99-bit words), 4 Kword ROM and 2 Kword writable control store
Internal data path	32 bits
Instruction buffer size	8-byte lookahead
Memory cache	8-KB, 2-way set associative

VAX Instruction Set	
Number of 32-bit registers	16
Number of basic operations	304
Number of priority interrupt levels	32
Number of addressing modes	9
Data types	Integer, floating-point, packed decimal, character string, variable-bit fields, and numeric strings

Main Memory	
Virtual address space	4 GB
Physical address space	16 MB total (64-KB chip memory), 32 MB total (256-KB chip memory)
Parity	8-bit error-correcting code (ECC) per 64-bit quadword
Memory cycle time	800 ns per 64-bit read (1,300 ns with single-bit errors), 1,400 ns per 64-bit write

CPU Address Translation Buffer	
Size	128 address translations
Typical hit ratio	97%

CPU Clocks	
Realtime clock	Programmable, crystal controlled, 0.01% accuracy, 1 μs resolution
Time-of-year clock	Includes battery backup for more than 100 hours

Console Subsystem	
Interface	LSI-11 microprocessor-controlled
Console drive	RX01 floppy-disk drive
Disk size	8 in
Storage capacity	256 KB per disk 3 KB per track 128 KB per sector
Transfer rate	50 KB/s

CPU Options	
CPU memory expansion	2-MB increments (64-KB chip memory), 8-MB increments (265-KB chip memory)
Communication interfaces	8-, 16-, and 24-line asynchronous interfaces, multifunction controller, and Ethernet controller
MA780 multiport memory	Two subsystems per CPU (4 MB total)
DR780 general-purpose interface	32-bit parallel data interface
DW780 UNIBUS adapter	4 total
H7112 memory battery backup	One per memory controller
FP780 floating-point accelerator	Single- and double-precision floating-point instructions and POLY, EMOD, and MULL instructions
KE780 floating-point	G and H extended-range floating-point microcode
KU780 user writable control store	2 KB (99-bit words)
CI780 adapter	Computer interconnect for VAXcluster
MASSBUS adapters	4 total; up to 8 high-speed disks or tapes per adapter

Operating Environment	
Temperature	15°C to 32°C (59°F to 90°F)
Relative humidity	20% to 80% (noncondensing)
Altitude	2,400 m (8,000 ft)
Heat dissipation	5350 kcal/h (21,230 Btu/h) (max.)

Processor Power Requirements	
ac line voltage tolerance	104-128 V, 60 Hz 191-233 V, 50 Hz 208-254 V, 50 Hz
Line frequency tolerance	59-61 Hz (60 Hz) 49-51 Hz (50 Hz)
Phases	3
ac power consumption	6,225 W

Physical Characteristics	
Weight	498 kg (1,100 lb)
Height	153.7 cm (60.5 in)
Width	118.1 cm (46.5 in)
Depth	76.2 cm (30 in)

Typical Power and Thermal Dissipation

Option	*Watts*	*Heat Dissipation*
Fully configured CPU cabinet	6,225	5,350 kcal/h (21,230 Btu/h)
Fully configured CPU expansion cabinet (H9602-HA, -HB)	2,000	1,720 kcal/h (6,820 Btu/h)
Fully configured UNIBUS expansion cabinet (H9652-MF, -MH)	2,000	1,720 kcal/h (6,820 Btu/h)
Fully configured multiport memory cabinet (MA780-JA, -JB)	1,800	1,550 kcal/h (6,140 Btu/h)
RH750 MASSBUS adapter	150	130 kcal/h (512 Btu/h)
UNIBUS adapter (DW780-AA, -AB)	300	260 kcal/h (1,024 Btu/h)
CI780 computer interconnect	242	210 kcal/h (833 Btu/h)
DR780 general-purpose interface	226	194 kcal/h (771 Btu/h)

Option	*Watts*	*Heat Dissipation*
FP780 floating-point accelerator	300	260 kcal/h (1,025 Btu/h)
KU780-A user writable control store	100	86 kcal/h (341 Btu/h)
DMF32 communication controller	55	47 kcal/h (188 Btu/h)
DEUNA Ethernet controller	70	60 kcal/h (239 Btu/h)
H7112 memory battery backup	25	22 kcal/h (85 Btu/h)

VAX-11/782 Dual-Processor

The CPU specifications apply to both processors.

Processor Type	
Microcontrol store instruction time	200 ns
Control store size	6-KB (99-bit words), 4 Kword ROM and 2 Kword user control store
Internal data path	32 bits
Instruction buffer size	8-byte lookahead
Memory cache	8 KB, 2-way set associative

VAX Instruction Set	
Number of 32-bit registers	16
Number of basic operations	304
Number of priority interrupt levels	32
Number of addressing modes	9
Data types	Integer, floating-point, packed decimal, character string, variable-bit fields, and numeric strings
Local memory	265 KB used for diagnostic programs only

Shared Memory	
Virtual address space	4 GB
Physical address space	4 MB
Parity	8-bit error-correcting code (ECC) per 64-bit quadword
Memory cycle time	800 ns per 64-bit read (1,300 ns with single-bit errors), 1,400 ns per 64-bit write

CPU Address Translation Buffer	
Size	128 address translations
Typical hit ratio	97%

CPU Clocks	
Realtime clock	Programmable, crystal-controlled, 0.01% accuracy, 1 μs resolution
Time-of-year clock	Includes battery backup for more than 100 hours

Console Subsystem	
Interface	LSI-11 microprocessor controlled
Console drive	RX01 floppy-disk drive
Disk size	8 in
Storage capacity	256 KB per disk 3 KB per track 128 bytes per sector
Transfer rate	50 KB/s

CPU Options	
Memory expansion	4 MB (16-KB chip memory) with second MA780 cabinet
Communication interfaces	8-, 16-, and 24-line asynchronous interfaces, multifunction controller, and Ethernet controller
H7112 memory battery backup	One per memory controller
FP782 floating-point accelerator	Two accelerators; single- and double-precision, F and D floating-point instructions and POLY, EMOD, and MULL instructions
KE780 floating-point	One per CPU; G and H extended-range floating-point microcode
UNIBUS and MASSBUS adapters	4 total per system; up to 8 high-speed disks or tapes per adapter

Operating Environment	
Temperature	15°C to 32°C (59°F to 90°F)
Relative humidity	20% to 80% (noncondensing)
Altitude	2,400 m (8,000 ft)
Heat dissipation	5,350 kcal/h (21,230 Btu/h) (max.)

Processor Power Requirements	
ac line voltage tolerance	104 to 128 V, 60 Hz 191 to 233 V, 50 Hz 208 to 254 V, 50 Hz
Line frequency tolerance	59 to 61 Hz (60 Hz) 49 to 51 Hz (50 Hz)
Phases	3
ac power consumption	6,225 W (max.)

Physical Characteristics*	
Weight	1,315 kg (2,900 lb)
Height	153.7 cm (60.5 in)
Width	301.6 cm (118.75 in)
Depth	76.2 cm (30 in)

*Includes two processors and shared-memory cabinet

Typical Power and Thermal Dissipation

Option	*Watts*	*Heat Dissipation*
Fully configured primary CPU cabinet	6,225	5,350 kcal/h (21,230 Btu/h)
Fully configured attached CPU cabinet	5,000	3,964 kcal/h (15,700 Btu/h)
Fully configured multiport memory cabinet (MA780-KA, -KB)	1,800	1,550 kcal/h (6,140 Btu/h)
Fully configured CPU expansion cabinet (H9602-HA, -HB)	2,000	1,720 kcal/h (6,820 Btu/h)
Fully configured UNIBUS expansion cabinet (H9652-MF, -MH)	2,000	1,720 kcal/h (6,820 Btu/h)
RH780 MASSBUS adapter	150	130 kcal/h (512 Btu/h)
UNIBUS adapter (DW780-AA,-AB)	300	260 kcal/h (1,024 Btu/h)
FP782 floating-point accelerator	300	260 kcal/h (1,025 Btu/h)
H7112 memory battery backup	25	22 kcal/h (85 Btu/h)

▪ VAX-11/785 Processor

Processor Type

Microcontrol store instruction time	133 ns
Control store size	8 Kwords (0.5-KB ROM and 7.5-KB read-write) – including 1KB for user control store (99-bit words)
Internal data path	32 bits
System I/O transfer rate	13.3 MB (max.)
Instruction buffer size	8-byte lookahead
Memory cache	32 KB, 2-way set associative

VAX Instruction Set	
Number of 32-bit registers	16
Number of basic operations	304
Number of priority interrupt levels	32
Number of addressing modes	9
Data types	Integer, floating-point (G and H), packed decimal, character string, variable-bit fields, and numeric strings

Main Memory	
Virtual address space	4 GB
Physical address space	32 MB total (64-KB chip memory), 64 MB total (256-KB chip memory)
Parity	8-bit error-correcting code (ECC) per 64-bit quadword
Cycle time	600 ns per 64-bit read (700 ns with single-bit errors), 700 ns per 64-bit write

CPU Address Translation Buffer	
Size	128 address translations
Typical hit ratio	97%

CPU Clocks	
Realtime clock	Programmable, crystal-controlled, 0.01% accuracy, 1 μs resolution
Time-of-year clock	Includes battery backup for more than 100 hours

Console Subsystem	
Interface	LSI-11 microprocessor-controlled
Console drive	RX01 floppy-disk drive
Disk size	8 in
Storage capacity	256 KB per disk 3 KB per track 128 bytes per sector
Transfer rate	50 KB/s

CPU Options	
Memory expansion	2-MB increments (64-KB chip memory), 8-MB increments (256-KB chip memory)
MA780 multiport memory	Two per system (4 MBs total)
FP785 floating-point accelerator	Single- and double-precision instructions plus POLY, EMOD, and MULL instructions
DW780 UNIBUS adapter	4 total
MASSBUS adapters	4 total; up to 8 high-speed disks or tapes per adapter
DR780 general purpose interface	32-bit parallel data interface
Communication interfaces	8-, 16-, and 24-line asynchronous interfaces, multifunction controller, and Ethernet controller
H7112 memory battery backup	One per memory controller
UDA50 disk controller	One on first UNIBUS Two per each additional UNIBUS

Operating Environment	
Temperature	15°C to 32°C (59°C to 90°F)
Relative humidity	20% to 80% (noncondensing)
Altitude	2,400 m (8,000 ft)
Heat dissipation (max)	2,116 kcal/h (8,350 Btu/h)

Electrical Power Requirements*	
ac line voltage tolerance	104 to 128 V, 60 Hz 191 to 233 V, 50 Hz 208 to 254 V, 50 Hz
Line frequency tolerance	59 to 61 Hz (60 Hz) 49 to 51 Hz (50 Hz)
Phases	3
ac power consumption	2,500 W (max.)

Physical Characteristics*	
Weight	498 kg (1,100 lb)
Height	153.7 cm (60.5 in)
Width	118.1 cm (46.5 in)
Depth	76.2 cm (30 in)

*Includes CPU, one DW780, 2-MB memory (MS780-E)

Typical Power and Thermal Dissipation

Option	*Watts*	*Heat Dissipation*
Fully configured CPU expansion cabinet (H9652-HA, -HB)	2,000	1,722 kcal/h (6,820 Btu/h)
Fully configured UNIBUS expansion cabinet (H9652-MF, -MH)	2,000	1,720 kcal/h (6,820 Btu/h)
Fully configured multiport memory cabinet (MA780-JA, -JB)	1,800	1,550 kcal/h (6,140 Btu/h)
RH780 MASSBUS adapter	150	130 kcal/h (512 Btu/h)
UNIBUS adapter (DW780-AA, -AB)	300	260 kcal/h (1,024 Btu/h)
CI780 adapter	226	194 kcal/h (771 Btu/h)
FP785 floating-point accelerator	300	260 kcal/h (1,025 Btu/h)
KU780-A user writable control store	100	86 kcal/h (341 Btu/h)
DR780 general purpose interface	226	194 kcal/h (771 Btu/h)
DMF32 communication controller	55	47 kcal/h (188 Btu/h)
DEUNA Ethernet controller	70	60 kcal/h (239 Btu/h)
H7112 memory battery backup	25	22 kcal/h (85 Btu/h)

• VAX-8600 Processor

Processor Type	
Microcontrol store instruction time	80 ns
Control store size	8 KB (86-bit words)
Internal data path	32 bits
Instruction buffer size	8-byte lookahead
Memory cache	16 KB, 2-way set associative
Translation buffer	512 locations (one-way set associative)

VAX Instruction Set	
Number of 32-bit registers	16
Number of basic operations	304
Number of priority interrupt levels	32
Number of addressing modes	9
Data types	Integer, floating point (F, D, G, and H), packed decimal, character string, variable-bit fields, and numeric strings

Main Memory	
Virtual address capacity	4 GB
Capacity	32 MB max. (256-KB chip)
Parity	7-bit error-correcting code (ECC) per 32-bit longword
Cycle time	500 ns per 128-bit read

Console Subsystem	
Interface	T11 microprocessor-controlled
Console device	RL02 disk drive
Formatted capacity	10.4 MB
Transfer rate	512 KB/s (max.)
Access time	67.5 ms (avg)
Seek time	55 ms (avg)
Tracks per surface	512
Sectors per track	40
Bytes per sector	256

CPU options	
Memory expansion	4-MB increments (256-KB chip memory)
DB86 second SBI adapter	One per system
DW780 UNIBUS adapter	Two per CPU cabinet total
Communication interfaces	8-, 16-, and 24-line asynchronous interfaces, multifunction controller, and Ethernet controller

Operating Environment	
Temperature	15°C to 32°C (59°F to 90°F)
Relative humidity	20% to 80% (noncondensing)
Altitude	2,400 m (8,000 ft)
Heat dissipation	5,555 kcal/hr (22,000 Btu/hr)

Processor Power Requirements*	
ac line voltage tolerance	156 to 222 Vrms, 60 Hz 380 to 440 Vrms, 50 Hz
Line frequency tolerance	47 to 63 Hz
Line current	30 A at 156 Vrms 15 A at 312 Vrms
Phases	3
ac power consumption	6,500 W/10 kVA

Physical Characteristics	
Weight	782.1 kg (1,725 lb)
Height	153.7 cm (60.5 in)
Width	186 cm (73.25 in)
Depth	76.2 cm (30 in)

Typical Power and Thermal Dissipation

Option	*Watts*	*Heat Dissipation*
Fully configured UNIBUS expansion cabinet (H9652-MF, -MH)	2,000	1,720 kcal/h (6,820 Btu/h)
Fully configured SBI expansion cabinet (H9652-CA, -CB)		
DB86 second SBI adapter		
RH780 MASSBUS adapter	150	130 kcal/h (512 Btu/h)
UNIBUS adapter (DW780-AA, -AB)	300	260 kcal/h (1,024 Btu/h)

Glossary

abort:
An exception that occurs in the middle of an instruction and potentially leaves the registers and memory in an indeterminate state, such that the instruction cannot necessarily be restarted.

absolute indexed mode:
An indexed addressing mode in which the base operand specifier is addressed in absolute mode.

absolute mode:
In absolute mode addressing, the PC is used as the register in autoincrement deferred mode. The PC contains the address of the location containing the actual operand.

access control:
The mechanism in a node for screening inbound connect requests and verifying them against a local system account file. Access control is an optional Session Control function.

access mode:
1. Any of the four processor access modes in which software executes. Processor access modes are, in order from most to least privileged and protected: kernel (mode 0), executive (mode 1), supervisor (mode 2), and user (mode 3). When the processor is in kernel mode, the executing software has complete control of, and responsibility for, the system. When the processor is in any other mode, the processor is inhibited from executing privileged instructions. The Processor Status Longword contains the current access mode field. The operating system uses access modes to define protection levels for software executing in the context of a process. For example, the Executive runs in kernel and executive mode and is most protected. The command interpreter is less protected and runs in supervisor mode. The debugger runs in user mode and is no more protected than normal user programs. 2. See record access mode.

access type:
1. The way in which the processor accesses instruction operands. Access types are: read, write, modify, address, and branch. 2. The way in which a procedure accesses its arguments. 3. See also record access type.

access violation:
An attempt to reference an address that is not mapped into virtual memory or an attempt to reference an address that is not accessible by the current access mode.

account name:
A string that identifies a particular account used to accumulate data on a job's resource use. This name is the user's accounting charge number, not the user's UIC.

address:
A number used by the operating system and user software to identify a storage location. See also virtual address and physical address.

address access type:
The specified operand of an instruction is not directly accessed by the instruction. The address of the specified operand is the actual instruction operand. The context of the address calculation is given by the data type of the operand.

addressing mode:
The way in which an operand is specified; for example, the way in which the effective address of an instruction operand is calculated using the general registers. The basic general register addressing modes are called: register, register deferred, autoincrement, autoincrement deferred, autodecrement, displacement, and displacement deferred. In addition, there are six indexed addressing modes using two general registers, and literal mode addressing. The PC addressing modes are called: immediate (for register deferred mode using the PC), absolute (for autoincrement deferred mode using the PC), and branch.

address space:
The set of all possible addresses available to a process. Virtual address space refers to the set of all possible virtual addresses. Physical address space refers to the set of all possible physical addresses sent out on the SBI.

allocate a device:
To reserve a particular device unit for exclusive use. A user process can allocate a device only when that device is not allocated by any other process.

alphanumeric character:
An upper or lower case letter (A-Z, a-z), a dollar sign ($), an underscore (_), or a decimal digit (0-9).

alternate key:
An optional key within the data records in an indexed file, used by VAX-11 RMS to build an alternate index. See key (indexed files) and primary key.

American Standard Code for Information Interchange (ASCII):
A set of 8-bit binary numbers representing the alphabet, punctuation, numerals, and other special symbols used in text representation and communications protocol.

Ancillary Control process (ACP):
A process that acts as an interface between user software and an I/O driver. An ACP provides functions supplemental to those performed in the driver, such as file and directory management. Three examples of ACPs are the Files-11 ACP (F11ACP), the magnetic tape ACP (MTACP), and the networks ACP (NETACP).

area:
Areas are VAX-11 RMS maintained regions of an indexed file. They allow a user to specify placement and/or specific bucket sizes for particular portions of a file. An area consists of any number of buckets and there may be from 1 to 255 areas in a file.

Argument Pointer:
General register 12 (R12). By convention, AP contains the address of the base of the argument list for procedures initiated using the CALL instructions.

ASCII:
American Standard Code for Information Interchange. ASCII is a seven-bit-plus-parity code established by the American National Standards Institute to achieve compatibility among different computers and terminals.

assign a channel:
To establish the necessary software linkage between a user process and a device unit before a user process can transfer any data to or from that device. A user process requests the system to assign a channel and the system returns a channel number.

asynchronous record operation:
A mode of record processing in which a user program can continue to execute after issuing a record retrieval or storage request without having to wait for the request to be fulfilled.

Asynchronous System Trap:
A software-simulated interrupt to a user-defined service routine. ASTs enable a user process to be notified asynchronously with respect to its execution of the occurrence of a specific event. If a user process has defined an AST routine for an event, the system interrupts the process and executes the AST routine when that event occurs. When the AST routine exits, the system resumes the process at the point where it was interrupted.

Asynchronous System Trap level (ASTLVL):
A value kept in an internal processor register that is the highest access mode for which an AST is pending. The AST does not occur until the current access mode drops in priority (raises in numeric value) to a value greater than or equal to ASTLVL. Thus, an AST for an access mode will not be serviced while the processor is executing in a higher priority access mode.

authorization file:
See user authorization file.

autodecrement indexed mode:
An indexed addressing mode in which the base operand specifier uses autodecrement mode addressing.

autodecrement mode:
In autodecrement mode addressing, the contents of the selected register are decremented, and the result is used as the address of the actual operand for the instruction. The contents of the register are decremented according to the data type context of the register: 1 for byte, 2 for word, 4 for longword and floating, 8 for quadword and double floating.

autoincrement deferred indexed mode:
An indexed addressing mode in which the base operand specifier uses autoincrement deferred mode addressing.

autoincrement deferred mode:
In autoincrement deferred mode addressing, the specified register contains the address of a longword which contains the address of the actual operand. The contents of the register are incremented by 4 (the number of bytes in a longword). If the PC is used as the register, this mode is called absolute mode.

autoincrement indexed mode:
An indexed addressing mode in which the base operand specifier uses autoincrement mode addressing.

autoincrement mode:
In autoincrement mode addressing, the contents of the specified register are used as the address of the operand, then the contents of the register are incremented by the size of the operand.

balance set:
The set of all process working sets currently resident in physical memory. The processes whose working sets are in the balance set have memory requirements that balance with available memory. The balance set is maintained by the system swapper process.

base operand address:
The address of the base of a table or array referenced by index mode addressing.

base priority:
The process priority that the system assigns to a process when it is created. The scheduler never schedules a process below its base priority. The base priority can be modified only by the system manager of the process itself.

base register:
A general register used to contain the address of the first entry in a list, table, array or other data structure.

binding:
See linking.

bit string:
See variable-length bit field.

block:
1. The smallest addressable unit of data that the specified device can transfer in an I/O operation (512 contiguous bytes for most disk devices). 2. An arbitrary number of contiguous bytes used to store logically related status, control, or other processing information.

block I/O:
The set of VAX-11 RMS procedures that allow you direct access to the blocks of a file regardless of the organization.

bootstrap block:
A block in the index file on a system disk that contains a program that can load the operating system into memory and start its execution.

branch access type:
An instruction attribute which indicates that the processor does not reference an operand address, but that the operand is a branch displacement. The size of the branch displacement is given by the data type of the operand.

branch mode:
In branch addressing mode, the instruction operand specifier is a signed byte or word displacement. The displacement is added to the contents of the updated PC (which is the address of the first byte beyond the displacement), and the result is the branch address.

bucket:
A storage structure, consisting of from 1 to 32 blocks, used for building and processing relative and indexed files. A bucket contains one or more records or record cells. Buckets are the unit of contiguous transfer between VAX-11 RMS buffers and the disk.

buffered I/O:
See system buffered I/O.

bug check:
The operating system's internal diagnostic check. The system logs the failure and crashes the system.

byte:
A byte is eight contiguous bits starting on an addressable byte boundary. Bits are numbered from the right, 0 through 7, with bit 0 the low-order bit. When interpreted arithmetically, a byte is a 2's complement integer with significance increasing from bits 0 through bit 6. Bit 7 is the sign bit. The value of the signed integer is in the range -128 to 127 decimal. When interpreted as an unsigned integer, significance increases from bits 0 through 7 and the value of the unsigned integer is in the range 0 to 255 decimal. A byte can be used to store one ASCII character.

cache memory:
A small high-speed memory placed between slower main memory and the processor. A cache increases effective memory transfer rates and processor speed. It contains copies of data recently used by the processor, and fetches several bytes of data from memory in anticipation that the processor will access the next sequential series of bytes.

Carrier Sense Multiple Access With Collision Detection (CSMA/CD):
A distributed channel-allocation procedure in which every station can receive the transmissions of all other stations. Each station awaits an idle channel before transmitting, and each station can detect overlapping transmissions by other stations.

CCITT:
The International Telegraph and Telephone Consultative Committee, the technical committee of the International Telecommunications Union (ITU), which is responsible for the development of recommendations regarding telecommunications, including data communications.

cell frame:
See stack frame.

cell instructions:
The processor instructions CALLG (Call Procedure with General Argument List) and CALLS (Call Procedure with Stack Argument List).

cell stack:
The stack, and conventional stack structure, used during a procedure call. Each access mode of each process context has one cell stack, and interrupt service context has one cell stack.

channel:
A logical path connecting a user process to a physical device unit. A user process requests the operating system to assign a channel to a device so the process can transfer data to or from that device.

character:
A symbol represented by an ASCII code. See also alphanumeric character.

character string:
A contiguous set of bytes. A character string is identified by two attributes, an address and a length. Its address is the address of the byte containing the first character of the string. Subsequent characters are stored in bytes of increasing addresses. The length is the number of characters in the string.

character string descriptor:
A quadword data structure used for passing character data (strings). The first word of the quadword contains the length of the character string. The second word can contain type information. The remaining longword contains the address of the string.

cluster:
1. A set of contiguous blocks that is the basic unit of space allocation of a Files-11 disk volume. 2. A set of pages brought into memory in one paging operation. 3. An event flag cluster. 4. A system configuration typically consisting of more than one VAX processor and one or more hierarchical storage controllers. To the user, the cluster appears as though it were one computer system.

command:
An instruction, generally an English word, typed by he user, at a terminal or included in a command file which requests the software monitoring a terminal or reading a command file to perform some well-defined activity. For example, typing the COPY command requests the system to copy the contents of one file into another file.

command file:
A file containing command strings. See also command procedure.

command interpreter:
Procedure-based system code that executes in supervisor mode in the context of a process to receive, syntax check, and parse commands typed by the user at a terminal or submitted in a command file.

command parameter:
The positional operand of a command delimited by spaces, such as a file specification, option or constant.

command procedure:
A file containing commands and data that the command interpreter can accept in lieu of the user typing the commands individually on a terminal.

command string:
A line (or set of continued lines), normally terminated by typing the carriage return key, containing a command and, optionally, information modifying the command. A complete command string consists of a command, its qualifiers, if any, and its parameters (file specifications, for example), if any, and their qualifiers, if any.

common:
A FORTRAN term for a program section that contains only data.

common event flag cluster:
A set of 32 event flags that enables cooperating process to post event notification to each other. Common event flag clusters are created as they are needed. A process can associate with up to two common event flag clusters.

compatibility mode:
A mode of execution that enables the central processor to execute non-privileged PDP-11 instructions. The operating system supports compatibility mode execution by providing an RSX-11M programming environment for an RSX-11M task image. The operating system compatibility mode procedures reside in the control region of the process executing a compatibility mode image. The procedures intercept calls to the RSX-11M Executive and convert them to the appropriate operating system functions.

condition:
An exception condition detected and declared by software. For example, see failure exception mode.

condition codes:
Four bits in the Processor Status Word that indicate the results of previously executed instructions.

condition handler:
A procedure that a process wants the system to execute when an exception condition occurs. When an exception condition occurs, the operating system searches for a condition handler and, if found, initiates the handler immediately. The condition handler may perform some action to change the situation that caused the exception condition and continue execution for the process that incurred the exception condition. Condition handlers execute in the context of the process at the access mode of the code that incurred the exception condition.

condition value:
A 32-bit quantity that uniquely identifies an exception condition.

context:
The environment of an activity. See also process context, hardware context, and software context.

context indexing:
The ability to index through a data structure automatically because the size of the data type is known and used to determine the offset factor.

context switching:
Interrupting the activity in progress and switching to another activity. Context switching occurs as one process after another is scheduled for execution. The operating system saves the interrupted process' hardware context in the hardware process control block (PCB) using the Save Process Control instruction, and loads another process' hardware PCB into the hardware context using the Load Process Context instruction, scheduling that process for execution.

continuation character:
A hyphen at the end of a command line signifying that the command string continues on to the next command line.

console:
The manual control unit integrated into the central processor. The console includes an LSI-11 microprocessor and a serial line interface connected to a hard copy terminal. It enables the operator to start and stop the system, monitor system operation, and run diagnostics.

console terminal:
The hard copy terminal connected to the central processor console.

control region:
The highest-addressed half of per-process space (the P1 region). Control region virtual addresses refer to the process-related information used by the system to control the process, such as: the kernel, executive, and supervisor stacks, the permanent I/O channels, exception vectors, and dynamically used system procedures (such as the command interpreter and RSX-11M programming environment compatibility mode procedures). The user stack is also normally found in the control region, although it can be reallocated elsewhere.

Control Region Base Register (P1BR):
The processor register, or the equivalent in a hardware process control block, that contains the base virtual address of a process control region page table.

Control Region Length Register (P1LR):
The processor register, or its equivalent in a hardware process control block, that contains the number of non-existent page table entries for virtual pages in a process control region.

copy-on-reference:
A method used in memory management for sharing data until a process accesses it, in which case it is copied before the access. Copy-on-reference allows sharing of the initial values of a global section whose pages have read/write access but contain pre-initialized data available to many processes.

counted string:
A data structure consisting of a byte-sized length followed by the string. Although a counted string is not used as a procedure argument, it is a convenient representation in memory.

current access mode:
The processor access modes of the currently executing software. The Current Mode field of the Processor Status Longword indicates the access mode of the currently executing software.

cylinder:
The tracks at the same radius on all recording surfaces of a disk.

D__ floating (point) datum:
A floating point datum consisting of 8 contiguous bytes (64 bits) starting on an arbitrary byte boundary. The value of the D__ floating datum is in the approximate range (+ or –) 0.29×10^{-38} to 1.7×10^{38}. The precision is approximately one part in 255 or typically sixteen decimal digits.

data base:
1. All the occurrences of data described by a data base management system. 2. A collection of related data structures.

data structure:
Any table, list, array, queue, or tree whose format and access conventions are well-defined for reference by one or more images.

data type:
In general, the way in which bits are grouped and interpreted,. In reference to the processor instructions, the data type of an operand identifies the size of the operand and the significance of the bits in the operand. Operand data types include: byte, word, longword, and quadword integer, floating and double floating, character string, packed decimal string, and variable-length bit field.

deferred echo:
Refers to the fact that terminal echoing does not occur until a process is ready to accept input entered by type ahead.

delta time:
A time value expressing an offset from the current date and time. Delta times are always expressed in the system as negative numbers whose absolute value is used as an offset from the current time.

demand zero page:
A page, typically of an image stack or buffer area, that is initialized to contain all zeros when dynamically created in memory as a result of a page fault. This feature eliminates the waste of disk space that would otherwise be required to store blocks (pages) that contain only zeros.

descriptor:
A data structure used in calling sequences for passing argument types, addresses and other optional information. See character string descriptor.

detached process:
A process that has no owner. The parent process of a tree of subprocesses. Detached processes are created by the job controller when a user logs on the system or when a batch job is initiated. The job controller does not own the user processes it creates; these processes are therefore detached.

device:
The general name for any physical terminus or link connected to the processor that is capable of receiving, storing, or transmitting data. Card readers, line printers, and terminals are examples of record-oriented devices. Magnetic tape devices and disk devices are examples of mass storage devices. Terminal line interfaces and interprocessor links are examples of communications devices.

device interrupt:
An interrupt received on interrupt priority level 16 through 23. Device interrupts can be requested only by devices, controllers, and memories.

device name:
The field in a file specification that identifies the device unit on which a file is stored. Device names also include the mnemonics that identify an I/O peripheral device in a data transfer request. A device name consists of a mnemonic followed followed by a controller identification letter (if applicable), followed by a unit number (if applicable), and ends with a colon (:).

device queue:
See spool queue.

device register:
A location in device controller logic used to request device functions (such as I/O transfers) and/or report status.

device unit:
One drive, and its controlling logic, of a mass storage device system. A mass storage system can have several drives connected to it.

diagnostic:
A program that tests logic and reports any faults it detects.

direct I/O:
An I/O operation in which the system locks the pages containing the associated buffer in memory for the duration of the I/O operation. The I/O transfer takes place directly from the process buffer. Contrast with system buffered I/O.

direct mapping cache:
A cache organization in which only one address comparison is needed to locate any data in the cache because any block of main memory data can be placed in only one possible position in the cache. Contrast with fully associative cache.

directory:
A file used to locate files on a volume that contains a list of the names (including type and version number) and their unique internal identifications.

directory name:
The field in a file specification that identifies the directory file in which a file is listed. The directory name begins with a left bracket ([or <) and ends with a right bracket (] or >).

displacement deferred indexed mode:
An indexed addressing mode in which the base operand specifier uses displacement deferred mode addressing.

displacement deferred mode:
In displacement deferred mode addressing, the specifier extension is a byte, word, or longword displacement. The displacement is sign extended to 32 bits and added to a base address obtained from the specified register. The result is the address of a longword which contains the address of the actual operand. If the PC is used as the register, the updated contents of the PC are used as the base address. The base address is the address of the first byte beyond the specifier extension.

displacement indexed mode:
An indexed addressing mode in which the base operand specifier uses displacement mode addressing.

displacement mode:
In displacement mode addressing, the specifier extension is a byte, word, or longword displacement. The displacement is sign extended to 32 bits and added to a base address obtained from the specified register. The result is the address of the actual operand. If the PC is used as the register, the updated contents of the PC are used as the base address. The base address is the address of the first byte beyond the specifier extension.

drive:
The electro-mechanical unit of a mass storage device system on which a recording medium (disk cartridge, disk pack, or magnetic tape reel) is mounted.

driver:
the set of code that handles physical I/O to a device.

dynamic access:
A technique in which a program switches from one record access mode to another while processing a file.

echo:
A terminal handling characteristic in which the characters typed by the terminal user on the keyboard are also displayed on the screen or printer.

effective address:
The address obtained after indirect or indexing modifications are calculated.

end node:
A topological description of a non-routing node (one that cannot forward packets intended for other nodes). An end node supports only a single active line.

entry mask:
A word whose bits represent the registers to be saved or restored on a subroutine or procedure call using the call and return instructions.

entry point:
A location that can be specified as the object of a call. It contains an entry mask and exception enables known as the entry point mask.

equivalence name:
The string associated with a logical name in a logical name table. An equivalence name can be, for example, a device name, another logical name, or a logical name concatenated with a portion of a file specification.

error logger:
A system process that empties the error log buffers and writes the error messages into the error file. Errors logged by the system include memory system errors, device errors and timeouts, and interrupts with invalid vector addresses.

escape sequence:
An escape is a transition from the normal mode of operation to a mode outside the normal mode. An escape character is the code that indicates the transition from normal to escape mode. An escape sequence refers to the set of character combinations starting with an escape character that the terminal transmits without interpretation to the software set up to handle escape sequences.

Ethernet:
A local area network protocol using a carrier sense multiple access with collision detection (CSMA/CD) scheme to arbitrate the use of a 10 megabits-per-second baseband coaxial cable.

event:
A change in process status or an indication of the occurrence of some activity that concerns an individual process or cooperating processes. An incident reported to the scheduler that affects a process' ability to execute. Events can be synchronous with the process' execution (a wait request), or they can be asynchronous (I/O completion). Some other events include: swapping; wake request; page fault.

event flag:
A bit in an event flag cluster that can be set or cleared to indicate the occurrence of the event associated with that flag. Event flags are used to synchronize activities in a process or among many processes.

event flag cluster:
A set of 32 event flags which are used for event posting. Four clusters are defined for each process: two process-local clusters, and two common event flag clusters. Of the process-local flags, eight are reserved for system use.

exception:
An event detected by the hardware (other than an interrupt or jump, branch, case, or call instruction) that changes the normal flow of instruction execution. An exception is always caused by the execution of an instruction or set of instructions (whereas an interrupt is caused by an activity in the system independent of the current instruction). There are three types of hardware exceptions: traps, faults, and aborts. Examples are: attempts to execute a privileged or reserved instruction, trace faults, compatibility mode faults, break-point instruction execution, and arithmetic faults such as floating point overflow, underflow, and divide by zero.

exception condition:
A hardware- or software-detected event other than an interrupt or jump, branch, case, or call instruction that changes the normal flow of instruction execution.

exception dispatcher:
An operating system procedure that searches for a condition handler when an exception condition occurs. If no exception handler is found for the exception or condition, the image that incurred the exception is terminated.

exception enables:
See trap enables.

exception vector:
See vector.

executable image:
An image that is capable of being run in a process. When run, an executable image is read from a file for execution in a process.

executive:
The generic name for the collection of procedures included in the operating system software that provides the basic control and monitoring functions of the operating system.

executive mode:
The second most privileged processor access mode (mode 1). The record management services (RMS) and many of the operating system's programmed service procedures execute in executive mode.

exit:
An image exit is a rundown activity that occurs when image execution terminates either normally or abnormally. Image rundown activities include deassigning I/O channels and disassociation of common event flag clusters. Any user- or system-specified exit handlers are called.

extended attribute block (XAB):
An RMS user data structure that contains additional file attributes beyond those expressed in the file access block (FAB), such as boundary types (aligned on cylinder, logical block number, virtual block number) and the protection information.

extension:
The amount of space to allocate at the end of a file each time a sequential write exceeds the allocated length of the file.

extent:
The contiguous area on a disk containing a file or a portion of a file. Consists of one or more clusters.

F__ floating (point) datum:
A floating point datum consisting of 4 contiguous bytes (32 bits) starting on an arbitrary byte boundary. The value of the F__ floating datum is in the approximate range (+ or –) 0.29×10^{-38} to 1.7×10^{38}. The precision is approximately one part in 2^{23} or typically seven decimal digits.

failure exception mode:
A mode of execution selected by a process indicating that it wants an exception condition declared if an error occurs as the result of a system service call. The normal mode is for he system service to return an error status code for which the process must test.

fault:
A hardware exception condition that occurs in the middle of an instruction and that leaves the registers and memory in a consistent state, such that elimination of the fault and restarting the instruction will give correct results.

field:
1. See variable-length bit field. 2. A set of contiguous bytes in a logical record.

file:
An organized collection of related items (records) maintained in an accessible storage area, such as disk or tape.

file access block (FAB):
An RMS user data structure that represents a request for data operations related to the file as a whole, such as OPEN, CLOSE, or CREATE.

file header:
A block in the index file describing a file on a Files-11 disk structure. The file header identifies the locations of the file's extents. There is a file header for every file on the disk.

file name:
The field preceding a file type in a file specification that contains a 1- to 9-character logical name for a file.

filename extension:
See file type.

file organization:
The physical arrangement of data in the file. You select the specific organization from those offered by VAX-11 RMS, based on your individual needs for efficient data storage and retrieval. See indexed file organization, relative file organization, and sequential file organization.

FILES-11:
The name of the on-disk structure used by the RSX-11, IAS, and VAX/VMS operating systems. Volumes created under this structure are transportable between these operating systems.

file specification:
A unique name for a file on a mass storage medium. It identifies the node, the device, the directory name, the file name, the file type, and the version number under which a file is stored.

file system:
A method of recording, cataloging, and accessing files on a volume.

file type:
The field in a file specification that is preceded by a period or dot (.) and consists of a zero- to three-character type identification. By convention, the type identifies a generic class of files that have the same use or characteristics, such as ASCII text files, binary object files, etc.

fixed control area:
An area associated with a variable length record available for controlling or assisting record access operations. Typical uses include line numbers and printer format control information.

fixed-length record format:
Property of a file in which all records are of the same size. This format provides simplicity in determining the exact location of a record in the file and eliminates the need to prefix a record size field to each record.

floating (point) datum:
A numeric data type in which the number is represented by a fraction (less than 1 and greater than or equal to ½) multiplied by 2 raised to a power. There are four floating point data types: F__ floating (4 bytes), D__ floating (8 bytes), G__ floating (8 bytes), and H__ floating (16 bytes).

foreign volume:
Any volume other than a Files-11 formatted volume which may or may not be file structured.

fork process:
A dynamically created system process such as a process that executes device driver code or the timer process. Fork processes have minimal context. Fork processes are scheduled by the hardware rather than by the software. The timer process is dispatched directly by software interrupt. I/O driver processes are dispatched by a fork dispatcher. fork processes execute at software interrupt levels and are dispatched for execution immediately. Fork processes remain resident until they terminate.

frame pointer:
General register 13 (R13). By convention, FP contains the base address of the most recent call frame on the stack.

fully associative cache:
A cache organization in which any block of data from main memory can be placed anywhere in the cache. Address comparison must take place against each block in the cache to find any particular block. Contrast with direct mapping cache.

G__ floating (point) datum:
A floating point datum consisting of 8 contiguous bytes (64 bits) starting on an arbitrary byte boundary. The value of the G__ floating datum is in the approximate range (+ or 1) 0.56×10^{-308} to 0.9×10^{308}. The precision is approximately one part in 2^{52} or typically fifteen decimal digits.

gateway:
A module, or set of modules, that transforms the conventions of one network into the conventions of another.

general register:
Any of the sixteen 32-bit registers used as the primary operands of the native mode instructions. The general registers include 12 general purpose registers which can be used as accumulators, as counters, and as pointers to locations in main memory, and the Frame Pointer (FP), Argument Pointer (AP), Stack Pointer (SP), and Program Counter (PC) registers.

generic device name:
A device name that identifies the type of device but not a particular unit, a device name in which the specific controller and/or unit number is omitted.

giga:
Metric term used to represent the number 1 followed by nine zeros.

global page table:
The page table containing the master page table entries for global sections.

global section:
A data structure (e.g., FORTRAN global common) or shareable image section potentially available to all processes in the system. Access is protected by privilege and/or group number of the UIC.

global symbol:
A symbol defined in a module that is potentially available for reference by another module. The linker receives (matches references with definitions) global symbols. Contrast with local symbol.

global symbol table (GST):
In a library, an index of strongly defined global symbols used to access the modules defining the global symbols. The linker will also put global symbol tables into an image. For example, the linker appends a global symbol table to executable images that are intended to run under the symbolic debugger, and it appends a global symbol table to all shareable images.

group:
1. A set of users who have special access privileges to each other's directories and files within those directories (unless protected otherwise), as in the context "system, owner, group, world," where group refers to all members of a particular owner's group. 2. A set of jobs (processes and their subprocesses) who have access privileges to a group's common event flags and logical name tables, and may have mutual process controlling privileges, such as scheduling, hibernation, etc.

group number:
The first number in a User Identification Code (UIC).

H__ floating (point) datum:
A floating point datum consisting of 16 contiguous bytes (128 bits) starting on an arbitrary byte boundary. The value of the H__ floating datum is in the approximate range (+ or –) 0.84×10^{-4932} to 0.59×10^{4932}. The precision is approximately one part in 2^{112} or typically 33 decimal digits.

hardware context:
The values contained in the following registers while a process is executing the Program Counter (PC), the Processor Status Longword (PSL), the 14 general registers (R0 through R13), the four processor registers (P0BR, P0LP, P1BR, and P1LR) that describe the process virtual address space, the Stack Pointer (SP) for the current access mode in which the processor is executing; plus the contents to be loaded in the Stack Pointer for every access mode other than the current access mode. While a process is executing, its hardware context is continually being updated by the processor. While a process is not executing, its hardware context is stored in its hardware PCB.

hierarchical network:
A computer network in which computers perform process-control functions at several levels. The computers at each level are specially suited to perform the functions of that level. In a laboratory application, for example, computers at the lowest level collect data, which they transmit to the computers at th next highest level. Here, the collected data is reduced for analysis, and sent up to th next level of the hierarchical configuration.

hardware process control block (PCB):
A data structure known to the processor that contains the hardware context when a process is not executing. A process' hardware PCB resides in its process header.

hibernation:
A state in which a process is inactive, but known to the system with all of its current status. A hibernating process becomes active again when a wake request is issued. It can schedule a wake request before hibernating, or another process can issue its wake request. A hibernating process also becomes active for the time sufficient to service any AST it may receive while it is hibernating. Contrast with suspension.

home block:
A block in the index file that contains the volume identification, such as volume label and protection.

image:
An image consists of procedures and data that have been bound together by the linker. There are three types of images, executable, shareable, and system.

image activator:
A set of system procedures that prepare an image for execution. The image activator establishes the memory management data structures required both to map the image's virtual pages to physical pages and to perform paging.

image exit:
See exit.

image I/O segment:
That portion of the control region that contains the RMS internal file access blocks (IFAB) and I/O buffers for the image currently being executed by a process.

image name:
The file name of the file in which an image is stored.

image privileges:
The privileges assigned to an image when it is linked. See process privileges.

image section (isect):
A group of program sections (isects) with the same attributes (such as read-only access, read/write access, absolute, relocatable, etc.) that is the unit of virtual memory allocation for an image.

immediate mode:
In immediate mode addressing, the PC is used as the register in autoincrement mode addressing.

index:
The structure which allows retrieval of records in an indexed file by key value. See key (indexed files).

index file:
The file on a Files-11 volume that contains the access information for all files on the volume and enables the operating system to identify and access the volume.

index file bit map:
A table in the index file of a Files-11 volume that indicates which file headers are in use.

index register:
A register used to contain an address offset.

indexed addressing mode:
In indexed mode addressing, two registers are used to determine the actual instruction operand: an index register and a base operand specifier. The contents of the index register are used as an index (offset) into a table or array. The base operand specifier supplies the base address of the array (the base operand address or BOA). The address of the actual operand is calculated by multiplying the contents of the index register by the size (in bytes) of the actual operand and adding the result to the base operand address. The addressing modes resulting from index mode addressing are formed by adding the suffix "indexed" to the addressing mode of the base operand specifier: register deferred indexed, autoincrement indexed, autoincrement deferred indexed (or absolute indexed), autodecrement indexed, displacement indexed, and displacement deferred indexed.

indexed file organization:
A file organization which allows random retrieval of records by key values and sequential retrieval of records in sorted order by key value. See key (indexed files).

indirect command file:
See command procedure.

input stream:
The source of commands and data. One of: the user's terminal, the batch stream, or an indirect command file.

instruction buffer:
A buffer in the processor used to contain bytes of the instruction currently being decoded and to pre-fetch instructions in the instruction stream. The control logic continuously fetches data from memory to keep the buffer full.

interleaving:
Assigning consecutive physical memory address alternately between two memory controllers.

interlocked:
The property of a read followed by a write to the same datum with no possibility of an intervening reference by a second processor or I/O device. Examples are the Branch on Bit interlocked and Add Aligned Work Interlocked instructions.

interprocess communication facility:
A common event flag, mailbox, or global section used to pass information between two or more processes.

interrecord gap:
A blank space deliberately placed between data records on the recording surface of a magnetic tape.

interrupt:
An event other than an exception or branch, jump, case, or call instruction that changes the normal flow of instruction execution. Interrupts are generally external to the process executing when the interrupt occurs. See also device interrupt, software interrupt, and urgent interrupt.

interrupt priority level (IPL):
The interrupt level at which the processor executes when an interrupt is generated. There are 31 possible interrupt priority levels. IPL 1 is lowest, 31 highest. The levels arbitrate contention for processor service. For example, a device cannot interrupt the processor if the processor is currently executing at an interrupt priority level greater than the interrupt priority level of the device's interrupt service routine.

interrupt service routine:
The routine executed when a device interrupt occurs.

interrupt stack:
The system-wide stack used when executing in interrupt service context. At any time, the processor is either in a process context executing in user, supervisor, executive or kernel mode, or in system-wide interrupt service context operating with kernel privileges, as indicated by the interrupt stack and current mode bits in the PSL. The interrupt stack is not context switched.

interrupt stack pointer:
The stack pointer for the interrupt stack. Unlike the stack pointers for process context stacks, which are stored in the hardware PCB, the interrupt stack pointer is stored in an internal register.

interrupt vector:
See vector.

I/O driver:
See driver.

I/O function:
An I/O operation that is interpreted by the operating system and typically results in one or more physical I/O operations.

I/O function code:
A 6-bit value specified in a Queue I/O Request system service that describes the particular I/O operation to performed (e.g., read, write, rewind).

I/O function modifier:
A 10-bit value specified in a Queue I/O Request system service that modifies an I/O function code (e.g., read terminal input no echo).

I/O lockdown:
The state of a page such that it cannot be paged or swapped out of memory until any I/O in progress to that page is completed.

I/O rundown:
An operating system function in which the system clears up any I/O in progress when an image exits.

I/O space:
The region of physical address space that contains the configuration registers, and device control/status and data registers.

I/O status block:
A data structure associated with the Queue I/O Request system service. This service optionally returns a status code, number of bytes transferred, and device- and function-dependent information in an I/O status block. It is not returned from the service call, but filled in when the I/O request completes.

ISO Reference Model:
The International Standards Organization Reference Model for Open System Interconnection, ISO draft proposal DP7498. A proposed international standard for network architectures that defines a seven-layer model, specifying services and protocols for each layer.

job:
1. A job is the accounting unit equivalent to a process and the collection of all the subprocesses, if any, that it and its subprocesses create. Jobs are classified as batch and interactive. For example, the job controller creates an interactive job to handle a user's requests when the user logs onto the system and it creates a batch job when the symbiont manager passes a command input file to it.
2. A print job.

job controller:
The system process that establishes a job's process context, starts a process running the LOGIN image for the job, maintains the accounting record for the job, manages symbionts, and terminates a process and its subprocesses.

job queue:
A list of files that a process has supplied for processing by a specific device, for example, a line printer.

kernel mode:
The most privileged processor access mode (mode D). The operating system's most privileged services, such as I/O drivers and the pager, run in kernel mode.

key

indexed files: A character string, a packed decimal number, a 2- or 4-byte unsigned binary number, or a 2- or 4-byte signed integer within each data record in an indexed file. You define the length and location within the records; VAX-11 RMS used the key to build an index. See primary key, alternate key, and random access by key value.

relative files: The relative record number of each data record in a data file; VAX-11 RMS uses the relative record numbers to identify and access data records in a relative file in random access mode. See relative record number.

lexical function:
A command language construct that the command interpreter evaluates and substitutes before it performs expression analysis on a command string. Lexical functions return information about the current process, such as UIC or default directory; and about character strings, such as length or substring locations.

librarian:
A program that allows the user to create, update, modify, list, and maintain object library, image library, and assembler macro library files.

library file:
A direct access file containing one or more modules of the same module type.

limit:
The size or number of given items requiring system resources (such as mailboxes, locked pages, I/O requests, open files, etc.) that a job is allowed to have at any one time during execution, as specified by the system manager in the user authorization file. See also quota.

line number:
A number used to identify a line of text in a file processed by a text editor.

linker:
A program that reads one or more object files created by language processors and produces an executable image file, a shareable image file, or a system image file.

linking:
The resolution of external references between object modules used to create an image, the acquisition of referenced library routines, service entry points, and data for the image, and the assignment of virtual addresses to components of an image.

literal mode:
In literal mode addressing, the instruction operand is a constant whose value is expressed in a 6-bit field of the instruction. If the operand data type is byte, word, longword, quadword, or octaword, the operand is zero-extended and can express values in the range 0 through 63 (decimal). If the Operand data type is F__, D__, G__, or H__ floating, the 6-bit field is composed of two 3-bit fields, one for the exponent and the other for the fraction. The operand is extended to F__, D__, G__, or H__ floating format.

locality:
See program locality.

local symbol:
A symbol meaningful only to the module that defines it. Symbols not identified to a language processor as global symbols are considered to be local symbols. A language processor resolves (matches references with definitions) local symbols. They are not known to the linker and cannot be made available to another object module. They can, however, be passed through the linker to the symbolic debugger. Contrast with global symbol.

locate mode:
Technique used for a record input operation in which the data records are not copies from the I/O buffer. See move mode.

locking a page in memory:
Making a page in an image ineligible for either paging or swapping. A page stays locked in memory until it is specifically unlocked.

locking a page in the working set:
Making a page in an image ineligible for paging out of the working set for the image. The page can be swapped when the process is swapped. A page stays locked in a working set until it is specifically unlocked.

logical block number:
A number used to identify a block on a mass storage device. The number is a volume-relative address rather than the physical (device-oriented) address or its virtual (file-relative) address. The flocks that constitute the volume are labeled sequentially starting with logical block 0.

logical I/O function:
A set of I/O operations (e.g., read and write logical block) that allow restricted direct access to device level I/O operations using logical block addresses.

logical name:
A user-specified name for any portion or all of a file specification. For example, the logical name INPUT can be assigned to a terminal device from which a program reads data entered by a user. Logical name assignments are maintained in logical name tables for each process, each group, and the system. A logical name can be created and assigned a value permanently or dynamically.

logical name table:
A table that contains a seat of logical names and their equivalence names for a particular process, a particular group, or the system.

logical I/O functions:
A set of I/O functions that allow restricted direct access to device level I/O operations.

logical record:
A group of related fields treated as a unit.

longword:
Four contiguous bytes (32 bits) starting on an addressable byte boundary. bits are numbered from right to left, 0 through 31. The address of the longword is the address of the byte containing bit 0. When interpreted arithmetically, a longword is a 2's complement integer with significance increasing from bit 0 through bit 30. When interpreted as a signed integer, bit 31 is the sign bit. The value of the signed integer is in the range -2,147,483,658 to 2,147,483,647. When interpreted as an unsigned integer, significance increases from bit 0 to bit 31. The value of the unsigned integer is in the range 0 through 4,294,967,295.

macro:
A statement that requests a language processor to generate a predefined set of instructions.

mailbox:
A software data structure that is treated as a record-oriented device for general interprocess communication. Communication using a mailbox is similar to other forms of device-independent I/O. Senders perform a write to a mailbox, the receiver performs a read from that mailbox. Some system-wide mailboxes are defined: the error logger and OPCOM read from system-wide mailboxes.

main memory:
See main memory.

mapping window:
A subset of the retrieval information for a file that is used to translate virtual block numbers to logical block numbers.

mass storage device:
A device capable of reading and writing data on mass storage media such as disk pack or a magnetic tape reel.

member number:
The second number in a user identification code that uniquely identifies that code.

memory management:
The system functions that include the hardware a page mapping and protection and the operating system's image activator and pager.

Memory Mapping Enable (MME):
A bit in a processor register that governs address translation.

modify access type:
The specified operand of an instruction or procedure is read, and potentially modified and written, during that instruction's or procedure's execution.

module:
1. A portion of a program or program library, as in a *source module, object module,* or *image module.* 2. A board, usually made of plastic covered with an electrical conductor, on which logic devices (such as transistors, resistors, and memory chips) are mounted, and circuits connecting these devices are etched, as in a *logic module.*

Monitor Console Routine (MCR):
The command interpreter in an RSX-11 system.

mount a volume:
1. To logically associate a volume with the physical unit on which it is loaded (an activity accomplished by system software at the request of an operator). 2. To load or place a magnetic tape or disk pack on a drive and place the drive online (an activity accomplished by a system operator).

move mode:
Technique used for a record transfer in which the data records are copied between the I/O buffer and your program buffer for calculations or operations on the record. See locate mode.

multiplex:
Simultaneously transmit or receive two or more datastreams on a single channel.

mutex:
A semaphore that is used to control exclusive access to a region of code that can share a data structure or other resource. The mutex (mutual exclusion) semaphore ensures that only process at a time has access to the region of code.

name block (NAM):
An RMS user data structure that contains supplementary information used in parsing the specifications.

native image:
An image whose instructions are executed in native mode.

native mode:
processor's primary execution mode in which the programmed instructions are interpreted as byte-signed, variable-length instructions that operate on byte, word, longword, quadword, and octaword integer, F__, D__, G__, and H__ floating format, character string, packed decimal, and variable-length bit field data. The instruction execution mode other than compatibility mode.

network:
A collection of interconnected individual computer systems.

nibble:
The low-order or high-order four bits of a byte.

node:
An individual computer system in a network.

null process:
A small system process that is the lowest priority process in the system and takes one entire priority class. One function of the null process is to accumulate idle processor time.

numeric string:
A contiguous sequence of bytes representing up to 31 decimal digits (one per byte) and possibly a sign. The numeric string is specified by its lowest addressed location, its length, and its sign representation.

object module:
The binary output of a language processor such as the assembler or a compiler, which is used as input to the linker.

object time system (OTS):
See Run Time Procedure Library.

octaword:
Sixteen contiguous bytes (128 bits) starting on an addressable byte boundary. Bits are numbered from right to left, 0 to 127. An octaword is identified by the address of the byte containing the low-order bit (0).

offset:
A fixed displacement from the beginning of a data structure. system offsets for terms within a data structure normally have an associated symbolic name used instead of the numeric displacement. Where symbols are defined, programmers always reference the symbolic names in a data structure instead of using the numeric displacement.

opcode:
The pattern of bits within an instruction that specify the operations to be performed.

operand specifier:
The pattern of bits in an instruction that include the addressing mode, a register and/or displacement, which, taken together, identify an instruction operand.

operand specifier type:
The access type and data type of an instruction's operand(s). For example, the test instructions are of read access type, since they only read the value of the operand. The operand can be of byte, word, or longword data type, depending on whether the opcode is for the TSTB (test byte), TSTW (test word), or TSTL (test longword) instruction.

Operator Communication Manager (OPCOM):
A system process that is always active. OPCOM receives input from a process that wants to inform an operation of a particular status or condition, passes a message to the operator, and tracks, the message.

operator's console:
Any terminal identified as a terminal attended by a system operator.

owner:
In the context "system, owner, group, world," an owner is the particular member (of a group) to which a file, global section, mailbox, or event flag cluster belongs.

owner process:
The process (with the exception of the job controller) or subprocess that created a subprocess.

packed decimal:
A method of representing a decimal number by storing a pair of decimal digits in one byte, taking advantage of the fact that only four bits are required to represent the numbers 0 through 9.

packed decimal string:
A contiguous sequence of up to 16 bytes interpreted as a string of nibbles. Each nibble represents a digit except the low-order nibble of the highest addressed byte, which represents the sign. The packed decimal staring is specified by its lowest addressed location and the number of digits.

packet:
A unit of data to be routed from a source node to a destination node. When stripped of its route header and passed to the End-to-End Communication Layer, a packet becomes a datagram.

packet level:
Level 3 of the CCITT X.25 recommendation, which defines the packet format and control procedures for the exchange of packets.

page:
1. A set of 512 contiguous byte locations used as the unit of memory mapping and protection. 2. The data between the beginning of file and a page marker, between two markers, or between a marker and the end of file.

page fault:
An exception generated by a reference to a page which is not mapped into a working set.

page fault cluster size:
The number of pages read in on a page fault.

page frame number (PFN):
The address of the first byte of a page in physical memory. The high-order 21 bits of the physical address of the base of a page.

page marker:
A character or characters (generally a form feed) that separates pages in a file that is processed by a text editor.

pager:
A set of kernel mode procedures that executes as the result of a page fault. The pager makes the page for which the fault occurred available in physical memory so that the image can continue execution. The pager and the image activator provide the operating system's memory management functions.

page table entry (PTE):
The data structure that identifies the location and status of a page of virtual address space. When a virtual page is in memory, the PTE contains the page frame number needed to map the virtual page to a physical page. When it is not in memory, the page table entry contains the information needed to locate the page on secondary storage (disk).

paging:
The action of bringing pages of an executing process into physical memory when referenced. When a process executes, all of its pages are said to reside in virtual memory. Only the actively used pages, however, need to reside in physical memory. The remaining pages can reside on disk until they are needed in physical memory. In this system, a process is paged only when it references more pages than it is allowed to have in its working set. When the process refers to a page not in its working set, a page fault occurs. This causes the operating system's pager to read in the referenced page if it is on disk (and, optionally, other related pages depending on a cluster factor), replacing the least recently faulted pages as needed. A process pages only against itself.

parameter:
See command parameter.

per-process address space:
See process address space

physical address:
The address used by hardware to identify a location in physical memory or on directly addressable secondary storage devices such as a disk. A physical memory address consists of a page frame number and the number of a byte within the page. A physical disk block address consists of a cylinder or track and sector number.

physical address space:
The set of all possible 30-bit physical addresses that can be used to refer to locations in memory (memory space) or device registers (I/O space).

physical block:
A block on a mass storage device referred to by its physical (device-oriented) address rather than a logical (volume-relative) or virtual (file-relative) address.

physical I/O functions:
A set of I/O functions that allow access to all device level I/O operations except maintenance mode.

physical memory:
the memory modules connected to the SBI that are used to store: 1) instructions that the processor can directly fetch and execute, and 2) any other data that a processor is instructed to manipulate. Also called main memory.

position-dependent code:
Code that can execute properly only in the locations in virtual address space that are assigned to it by the linker.

position-independent code:
Code that can execute properly without modification wherever it is located in virtual address space, even if its location is changed after it has been linked. Generally, this code uses addressing modes that form an effective address relative to the PC.

primary key:
The mandatory key within the data records of an indexed file, used by VAX-11 RMS to determine the placement of records within the file and to build the primary index. See key (indexed files) and alternate key.

primary vector:
A location that contains the starting address of a condition handler to be executed when an exception condition occurs. If a primary vector is declared, that condition handler is the first handler to be executed.

private section:
An image section of a process that is not shareable among processes. See also global section.

privilege:
See process privilege, user privilege, and image privilege.

privileged instructions:
In general, any instructions intended for use by the operating system or privileged system programs. In particular, instructions that the processor will not execute unless the current access mode is kernel mode (e.g., HALT, SVPCTX, LDPCTX, MTPR, and MFPR).

procedure:
1. A routine entered via a Call instruction. 2. See command procedure.

process:
The basic entity scheduled by the system software that provides the context in which an image executes. A process consists of an address space and both hardware and software context.

process address space:
See process space.

process context:
The hardware and software contexts of a process.

process control block (PCB):
A data structure used to contain process context. The hardware PCB contains the hardware context. The software PCB contains the software context, which includes a pointer to the hardware PCB.

process header:
A data structure that contains the hardware PCB, accounting and quota information, process section table, working set list, and the page tables defining the virtual layout of the process.

process header slots:
That portion of the system address space in which the system stores the process headers for the processes in the balance set. The number of process header slots in the system determines the number of processes that can be in the balance seta at any one time.

process identification (PID):
The operating system's unique 32-bit binary value assigned to a process.

process I/O segment:
That portion of a process control region that contains the process permanent RMS internal file access block for each open file, and the I/O buffers, including the command interpreter's command buffer and command descriptors.

process name:
A 1- to 15-character ASCII string that can be used to identify processes executing under the same group number.

processor register:
A part of the processor used by the operating system software to control the execution states of the computer system. The include the system base and length registers, the program and control region base and length registers, the system control block base register, the software interrupt request register, and many more.

Processor Status Longword (PSL):
A system programmed processor register consisting of a word of privileged processor status and the PSW. the privileged processor status information includes: the current IPL (interrupt priority level), the previous access mode, the current access mode, the interrupt stack bit, the trace fault pending bit, and the compatibility mode bit.

Processor Status Word (PSW):
The low-order word of the Processor Status Longword. Processor status information includes the condition codes (carry, overflow, zero, negative), the arithmetic trap enable bits (integer overflow, floating underflow), and the trace enable bit.

process page tables:
The page tables used to describe process virtual memory.

process priority:
The priority assigned to a process for scheduling purposes. The operating system recognizes 32 levels of process priority, where 0 is low and 31 high. Levels 16 through 31 re used for time-critical processes. The system does not modify the priority of a time-critical process (although the system manager or process itself may). Levels 0 through 15 are used for normal processes. The system may temporarily increase he priority of a normal process based on the activity of the process.

process privileges:
The privileges granted to a process by the system, which are a combination of user privileges and image privileges. They include, for example, the privilege to affect other processes associated with the same group as the user's group, affect any process in the system regardless of UIC, set process swap mode, create permanent event flag clusters, create another process, create a mailbox, and perform direct I/O to a file-structured device.

process section:
See private section.

process space:
The lowest-addressed half of virtual address space, where per-process instructions and data reside. Process space is divided into a program region and a control region.

Program Counter (PC):
General register 15 (R15). At the beginning of an instruction's execution, the PC normally contains the address of a location in memory from which the processor will fetch the next instruction it will execute.

program locality:
A characteristic of a program that indicates how close or far apart the references to locations in virtual memory are over time. A program with a high degree of locality does not refer to many widely scattered virtual addresses in a short period of time.

programmer number:
See member number.

program region:
The lowest-addressed half of process address space (P0 space). The program region contains the image currently being executed by the process and other user code called by the image.

Program region Base Register (P0BR):
The processor register, or its equivalent in a hardware process control block, that contains the base virtual address of the page table entry for virtual page number 0 in a process program region.

Program region Length Register (P0LR):
The processor register, or its equivalent in a hardware process control block, that contains the number of entries in the page table for a process program region.

program section (psect):
A portion of a program with a given protection and set of storage management attributes. Program sections that have the same attributes are gathered together by the linker to form an image section.

project number:
See group number or account number.

pure code:
See re-entrant code.

quadword:
Eight contiguous bytes (64 bits) starting on an addressable byte boundary. Bits are numbered from right to left, 0 to 63. A quadword is identified by the address of the byte containing the low-order bit (bit 0). When interpreted arithmetically, a quadword is a 2's complement integer with significance increasing from bit 0 to bit 62. Bit 63 is used as the sign bit. The value of the integer is in the range -2^{63} to $2^{63}-1$.

qualifier:
A portion of a command string that modifies a command verb or command parameter by selecting one of several options. A qualifier, if present, follows the command verb or parameter to which it applies and is in the formal "/qualifier option." For example, in the command string "PRINT filename/COPIES:3," the COPIES qualifier indicates that the user wants three copies of a given file printed.

queue:
1. n. A circular, doubly-linked list. See system queues. v. To make an entry in a list or table, perhaps using the INSQUE instruction. 2. See job queue.

queue priority:
The priority assigned to a job placed in a spooler queue or a batch queue.

quota:
The total amount of a system resource, such as CPU time, that a job is allowed to use in an accounting period, as specified by the system manager in the user authorization file. See also limit.

random access by key:
Indexed files only: Retrieval of a data record in an indexed file by either a primary or alternate key within the data record. See key (indexed files).

random access by record's file address:
The retrieval of a record by its unique address, which is provided to the program by RMS. This method of access is the only means of randomly accessing a sequentially organized file containing variable length records.

random access by relative record number:
Retrieval of a record by its relative record number. See relative record number. For relative files, random access by relative record number is synonymous with random access by key. See random access by key (relative files only).

read access type:
An instruction or procedure operand attribute indicating that the specified operand is only read during instruction or procedure execution.

record:
A set of related data that your program treats as a unit.

record access block (RAB):
An RMS user data structure that represents a request for a record access stream. A RAB relates to operations on the records within a file, such as UPDATE, DELETE, or GET.

record access mode:
The method used in RMS for retrieving and storing records in a file. One of three methods: sequential, random, and record's file address.

record blocking:
The technique of grouping multiple records into a single block. On magnetic tape, an IRG is placed after the block rather than after each record. This technique reduces the number of I/O transfers required to read or write the data, and, in addition (for magnetic tape), increases the amount of usable storage area. Record blocking also applies to disk files.

record cell:
A fixed-length area in a relative file that can contain a record. The concept of fixed-length record cells lets VAX-11 RMS directly calculate the record's actual position in the file.

record format:
The way a record physically appears on the recording surface of the storage medium. The record format defines the method for determining record length.

record length:
The size of a record; that is, the number of bytes in a record.

record locking:
A facility that prevents access to a record by more than one record stream or process until the initiating record stream or process releases the record.

Record Management Services:
A set of operating system procedures that are called by programs to process files and records within files. RMS allows programs to issue READ and WRITE requests at the record level (record I/O) as well as read and write blocks (block I/O). RMS is an integral part of the system software. RMS procedures run in executive mode.

record-oriented device:
A device such as a terminal, line printer, or card reader, on which the largest unit of data a program can access in one I/O operation is the device's physical record.

record's file address:
The unique address of a record in a file, which is returned by RMS whenever a record is accessed, that allows records in disk files to be accessed randomly regardless of file organization. This address is valid only for the life of the file. If an indexed file is reorganized, then the RFA of each record will typically change.

re-entrant code:
Code that is never modified during execution. It is possible to let many users share the same copy of a procedure or program written as re-entrant code.

register:
A storage location in hardware logic other than main memory. See also general register, processor register, and device register.

register deferred indexed mode:
An indexed addressing mode in which the base operand specifier uses register deferred mode addressing.

register mode:
In register mode addressing, the contents of the specified register are used as the actual instruction operand.

relative file organization:
The arrangement of records in a file where each record occupies a cell of equal length within a bucket. Each cell is assigned a successive number, called a relative record number, which represents the cell's position relative to the beginning of the file.

relative record number:
An identification number used to specify the position of a record cell relative to the beginning of the file, used as the key during random access by key mode to relative files.

remote node:
to a node, any other network node.

resource:
A physical part of the computer system such as a device or memory, or an interlocked data structure such as a mutex. Quotas and limits control the use of physical resources.

resource wait mode:
An execution state in which a process indicates that it will wait until a system resource becomes available when it issues a service request requiring a resource. If a process wants notification when a resource is not available, it can disable resource wait mode during program execution.

return status code:
See status code.

RMS-11:
A set of routines which is linked with compatibility mode programs, and provides similar functional capabilities to VAX-11 RMS. The file organizations and record formats used by RMS-11 are identical to those of VAX-11 RMS.

route through:
the process by which one or more intermediary node in the path between a source node and a destination node directs packats from the source node to the destination node. Routing nodes permit route-through Also called packet switching.

Run Time Library Procedure:
The collection of procedures available to native mode images at run time. These library procedures (such as trigonometric functions, etc.) are common to all native mode images, regardless of the language processor used to compile or assemble the program.

scatter/gather:
The ability to transfer in one I/O operation data from noncontiguous pages in memory to contiguous blocks on disk, or data from contiguous blocks on disk to noncontiguous pages in memory.

secondary storage:
Random access mass storage.

secondary vector:
A location that identifies the starting address of a condition handler to be executed when a condition occurs and the primary vector contains zero or the handler to which the primary vector points chooses not to handle the condition.

section:
A portion of process virtual memory that has common memory management attributes (protection, access, cluster factor, etc.). It is created from an image section, a disk file, or as the result of a Create Virtual Address Space system service. See global section, private section, image section, and program section.

self-relative queue:
A circularly linked list whose forward and backward links use the address of the entry in which they occur as the base address for the link displacement to the linked entry. Contrast with absolute addresses used to link a queue.

sequential file organization:
A file organization in which records appear in the order in which they were originally written. The records can be fixed length or variable length.

sequential record access mode:
Record storage or retrieval which starts at a designated point in the file and continues in one-after-the-other fashion through the file. That is, records are accessed in the order in which they physically appear in the file.

server:
a module or set of modules in a layer that performs a well-defined service, like remote file access or gateway communication, on behalf of another module.

shareable image:
An image that has all of its internal references resolved, but which must be linked with an object module(s) to produce an executable image. A sharable image cannot be executed. A shareable image file can be used to contain a library of routines. A shareable image can be used to create a global section by the system manager.

shell process:
A predefined process that the job initiator copies to create the minimum context necessary to establish a process.

signal:
1. An electrical impulse conveying information. 2. The software mechanism used to indicate that an exception condition was detected.

slave terminal:
A terminal from which it is not possible to issue commands to the command interpreter. A terminal assigned to application software.

small process:
A system process that has no control region in its virtual address space and has an abbreviated context. Examples are the working set swapper and the null process. A small process is scheduled in the same manner as user processes, but must remain resident during its execution.

software context:
The context maintained by the operating system that describes a process. See software process control block (PCB).

software interrupt:
An interrupt generated on interrupt priority level 1 through 15, which can be requested only by software.

software process control block (PCB):
The data structure used to contain a process' software context. The operating system defines a software PCB for every process when the process is created. The software PCB includes the following kinds of information about the process: current state; storage address if it is swapped out of memory; unique identification of the process, and address of the process header (which contains the hardware PCB). The software PCB resides in system region virtual address space. It is not swapped with a process.

software priority:
See process priority and queue priority.

spooling—*Output spooling:* The method by which output to a low-speed peripheral device (such as a line printer) is placed into queues maintained on a high-speed device (such as a disk) to await transmission to the low-speed device. *Input spooling:* The method by which input from a low-speed peripheral (such as the card reader) is placed into queues maintained on a high-speed device (such as a disk) to await transmission to a job processing that input.

spool queue:
The list of files supplied by processes that are to be processed by a symbiont. For example, a line printer queue is a list of files to be printed on the line printer.

stack:
An area of memory set aside for temporary storage, or for procedure and interrupt service linkages. A stack uses the last-in, first-out concept. As items are added to ("put on") the stack, the stack pointer decrements. As items are retrieved from ("popped off") the stack, the stack pointer increments.

stack frame:
A standard data structure built on the stack during a procedure call, starting from the location addressed by the FP to lower addresses, and popped off during a return from procedure. Also called call frame.

stack pointer:
General register 14 (R14). SP contains the address of the top (lowest address) of the processor-defined stack. reference to SP will access one of the five possible stack pointers (kernel, executive, supervisor, user, or interrupt) depending on the value in the current mode and interrupt stack bits in the Processor Status Longword (PSL).

state queue:
A list of processes in a particular processing state. The scheduler uses state queues to keep track of process' eligibility to execute. They include: processes waiting for a common event flag, suspended processes, and executable processes.

status code:
A longword value that indicates the success or failure of a specific function. For example, system services always return a status code in RO upon completion.

store through:
See write through.

strong definition:
Definition of a global symbol that is explicitly available for reference by modules linked with the module in which the definition occurs. The linker always lists a global symbol with a string definition in the symbol portion of the map. The librarian always includes a global symbol with a strong definition in the global symbol table of a library.

strong reference:
A reference to a global symbol in an object module that requests the linker to report an error if it does not find a definition for the symbol during linking. If a library contains the definition, the linker incorporates the library module defining the global symbol into the image containing the strong reference.

subprocess:
A subsidiary process created by another process. the process that creates a subprocess is its owner. A subprocess receives resource quotas and limits from its owner. When an owner process is removed from the system, all its subprocesses (and their subprocesses) are also removed.

supervisor mode:
The third most privileged processor access mode (mode 2). The operating system's command interpreter runs in supervisor mode.

suspension:
A state in which a process in inactive, but known to the system. A suspended process becomes active again only when another process requests the operating system to resume it. Contrast with hibernation.

swap mode:
A process execution state that determines the eligibility of a process to be swapped out of the balance set. If process swap mode is disabled, the process working set is locked in the balance set.

swapping:
The method for sharing memory resources among several processes by writing an entire working set to secondary storage (swap out) and reading another working set into memory (swap in). For example, a process' working set can be written to secondary storage while the process is waiting for I/O completion on a slow device. it is brought back into the balance set when I/O completes. Contrast with paging.

switch:
See (command) qualifier.

symbiont:
A full process that transfers record-oriented data to or from a mass storage device. For example, an input symbiont transfers data from card readers to disks. An output symbiont transfers data from disks to line printers.

symbiont manager:
The function (in the system process called the job controller) that maintains spool queues, and dynamically creates symbiont processes to perform the necessary I/O operations.

symbol:
See local symbol, global symbol, and universal global symbol.

Synchronous Backplane Interconnect (SBI):
The part of the hardware that interconnects the processor, memory controllers, MASSBUS adapters, and the UNIBUS adapter.

synchronous record operation:
A mode of record processing in which a user program issues a record read or write request and then waits until that request is fulfilled before continuing to execute.

system:
In the context "system, owner, group world," the system refers to the group numbers that are used by the operating system and its controlling users, the system operators and system manager.

system address space:
See system space and system region.

System Base Register (SBR):
A processor register containing the physical address of the base of the system page table.

system buffered I/O:
An I/O operation, such as terminal or mailbox I/O, in which an intermediate buffer from the system buffer pool is used instead of a process-specified buffer. Contrast with direct I/O.

System Control Block (SCB):
The data structure in system space that contains all the interrupt and exception vectors known to the system.

System Control Block Base register (SCBB):
A processor register containing the base address of the system control block.

system device:
The random access mass storage device unit which the volume containing the operating system software resides.

system dynamic memory:
Memory reserved for the operating system to allocate as needed for temporary storage. For example, when an image issues an I/O request, system dynamic memory is used to contain the I/O request packet. Each process has a limit on the amount of system dynamic memory that can be allocated for its use at one time.

System Identification Register:
A processor register which contains the processor type and serial number.

system image:
the image that is read into memory from secondary storage when the system is started up.

System Length Register (SLR):
A processor register containing the length of the system page table in longwords, that is, the number of page table entries in the system region page table.

System Page Table:
The data structure that maps the system region virtual addresses, including the addresses used to refer to the process page tables. The system Page Table (SPT) contains one Page Table Entry (PTE) for each page of system region virtual memory. The physical base address of the SPT is contained in a register called the SBR.

system process:
A process that provides system-level functions. Any process that is part of he operating system. See also small process, fork process.

system programmer:
A person who designs and/or writes operating systems, or who designs and writes procedures or programs that provide general purpose services for an application system.

system queue:
A queue used and maintained by operating system procedures. See also state queues.

system region:
The third quarter of virtual address space. The lowest-addressed half of system space. Virtual addresses in the system region are shareable between processes. Some of the data structures mapped by system region virtual addresses are system entry vectors, the System Control Block (SCB), the System Page Table (SPT), and process page tables.

system services:
Procedures provided by the operating system that can be called by user processes.

system space:
The highest-addressed half of virtual address space. See also system region.

system virtual address:
A virtual address identifying a location mapped by an address in system space.

system virtual space:
See system space.

task:
An RSX-11/IAS term for a process and image bound together.

terminal:
The general name for those peripheral devices that have keyboards and video screens or printers. Under program control, a terminal enables people to type commands and data on he keyboard and receive messages on the video screen or printer. Examples of terminals are the LA36 DECwriter hard-copy terminal and VT100 video display terminal.

time-critical process:
A process assigned to a software priority level between 16 and 31, inclusive. The scheduling priority assigned to a time-critical process is never modified by the scheduler, although it can be modified by the system manager or process itself.

timer:
A system fork process that maintains the time of day and the date. It also scans for device timeouts and performs time-dependent scheduling upon request.

transfer address:
The address of the location containing a program entry point (the first instruction to execute).

translation buffer:
An internal processor cache containing translations for recently used virtual addresses.

trap:
An exception condition that occurs at the end of the instruction that caused the exception. The PC saved on the stack is the address of the next instruction that would normally have been executed. All software can enable and disable some of the trap conditions with a single instruction.

trap enables:
Three bits in the Processor Status Word that control the processor's action on certain arithmetic exceptions.

two's complement:
A binary representation for integers in which a negative number is one greater than the bit complement of the positive number.

two-way associative cache:
A cache organization which has two groups of directly mapped blocks. Each group contains several blocks for each index position in the cache. A block of data from main memory can go into any group at its proper index position. A two-way associative cache is a compromise between the extremes of fully associative and direct mapping cache organizations that takes advantage of the features of both.

type ahead:
A terminal handling technique in which the user can enter commands and data while the software is processing a previously entered command. the commands typed ahead are not echoed on the terminal until the command processor is ready to process them. they are held in a type ahead buffer.

unit record device:
A device such as a card reader or line printer.

universal global symbol:
A global symbol in a shareable image that can be used by modules linked with that shareable image. Universal global symbols are typically a subset of all the global symbols in a shareable image. When creating a shareable image, he linker ensures that universal global symbols remain available for reference after symbols have been resolved.

unwind the call stack:
To remove call frames from the stack by tracing back through nested procedure calls using the current contents of the FP register and he FP register contents stored on the stack for each call frame.

urgent interrupt:
An interrupt received on interrupt priority levels 24 through 31. These can be generated only by the processor the for the interval clock, serious errors, and power fail.

user authorization file:
A file containing an entry for every user that the system manager authorizes to gain access to the system. Each entry identifies the user name, password, default account, User Identification Code (UIC), quotas, limits, and privileges assigned to individuals who use the system.

user environment test package (UETP):
A collection of routines that verify that the hardware and software systems are complete, properly installed, and ready to be used.

User File Directory (UFD):
See directory.

User Identification Code (UIC):
The pair of numbers assigned to users and to files, global sections, common event flag clusters, and mailboxes that specifies the type of access (read and/or write access, and in the case of files, execute and/or delete access) available to the owners, group, world, and system. It consists of a group number and a member number separated by a comma.

user mode:
The least privileged processor access mode (mode 3). User processes and the Run Time Library procedures run in user mode.

user name:
The name that a person types on a terminal to log on to the system.

user number:
See member number.

user privileges:
The privileges granted a user by the system manager. See process privileges.

utility:
A program that provides a set of related general purpose functions, such as a program development utility (an editor, a linker, etc.), a file management utility (the copy or file format translation program), or operations management utility (disk backup/restore, diagnostic program, etc.).

value return registers:
The general registers R0 and R1 used by convention to return function values. These registers are not preserved by any called procedures. They are available as temporary registers to any called procedure. All other registers (R2, R3, ..., R11, AP, FP, SP, PC) are preserved across procedure calls.

variable-length bit field:
A set of 0 to 32 contiguous bits located arbitrarily with respect to byte boundaries. A variable bit field is specified by four attributes: 1) the address A of a byte, 2) the bit position P of the starting location of the bit field with respect to bit 0 of the byte at address A, 3) the size, in bits, of the bit field, and 4) whether the field is signed or unsigned.

variable-length record format:
A file format in which records are necessarily the same length.

variable with fixed-length control record format:
Property of a file in which records of variable-length contain an additional fixed control area capable of storing data that may have no bearing on the other contents of the record. Variable with fixed-length control record format is not applicable to indexed files.

VAX-11 Record Management Services (VAX-11 RMS):
The file and record access subsystem of the VAX/VMS operating system for VAX. VAX-11 RMS helps your application program process records within files, thereby allowing interaction between your application program and its data.

vector:
1. A interrupt or exception vector is a storage location known to the system that contains the starting address of a procedure to be executed when a given interrupt or exception occurs. The system devises separate vectors for each interrupting device controller and for classes of exceptions. Each system vector is a longword. 2. For exception handling, users can declare up to two software exception vectors (primary and secondary) for each of the four access modes. Each vector contains the address of a condition handler. 3. A one-dimensional array.

version number:
1. The field following the file type in a file specification. It begins with a period (.) and is followed by a number which generally identifies it as the latest file created of all files having the identical file specification but for version number. 2. The number used to identify the revision level of program.

virtual address:
A 32-bit integer identifying a byte "location" in virtual address space. The memory management hardware translates a virtual address to a physical address. The term "virtual address" may also refer to the address used to identify a virtual block on a mass storage device.

virtual address space:
The set of all possible virtual addresses that an image executing in the context of a process can use to identify the location of an instruction or data. The virtual address space seen by the programmer is a linear array of 4,294,967,296 (2^{32}) byte addresses.

virtual block:
A block on a mass storage device referred to by its file-relative address rather than its logical (volume-oriented) or physical (device-oriented) address. The first block in a file is always virtual block 1.

virtual I/O functions:
A set of I/O functions that must be interpreted by an ancillary control process.

virtual memory:
The set of storage locations in physical memory and on disk that are referred to by virtual addresses. From the programmer's viewpoint, the secondary storage locations appear to be locations in physical memory. The size of virtual memory in any system depends on the amount of physical memory available and the amount of disk storage used for non-resident virtual memory.

virtual page number:
The virtual address of a page of virtual memory.

volume:

Disks: An ordered set of 512-byte blocks. The basic medium that carries a Files-11 structure.

Magnetic tape: A reel of magnetic tape, which may contain a part of a file, a complete file, or more than one file.

volume set:
A collection of related volumes.

wait:
To become inactive. A process enters a process wait state when the process suspends itself, hibernates, or declares that it needs to wait for an event, resource, mutex, etc.

wake:
To activate a hibernating process. A hibernating process can be awakened by another process or by the timer process, if the hibernating process or another process scheduled a wake-up call.

weak definition:
Definition of a global symbol that is not explicitly available for reference by modules linked with the module in which the definition occurs. The librarian does not include a global symbol with a weak definition in the global symbol table of a library. Weak definitions are often used when creating libraries to identify those global symbols that are needed only if the module containing them is otherwise linked with a program.

weak reference:
A reference to a global symbol that requests the linker not to report an error or to search the default library's global symbol table to resolve the reference if the definition is not in the modules explicitly supplied to the linker. Weak references are often used when creating object modules to identify those global symbols that may not be needed at run time.

wild card:
A symbol, such as an asterisk, that is used in place of a file name, file type, directory name, or version number in a file specification to indicate "all" for the given field.

window:
See mapping window.

word:
Two contiguous bytes (16 bits) starting on an addressable byte boundary. Bits are numbered from the right, 0 through 15. A word is identified by the address of the byte containing bit 0. When interpreted arithmetically, a word is a 2's complement integer with significance increasing from bit 0 to bit 14. If interpreted as a signed integer, bit 15 is the sign bit. The value of the integer is in he range -32,768 to 32,767. When interpreted as an unsigned integer, significance increases from bit 0 through bit 15 and the value of the unsigned integer is in the range 0 through 65,535.

working set:
The set of pages in process space to which an executing process can refer without incurring a page fault. The working set must be resident in memory for the process to execute. The remaining pages of that process, if any, are either in memory and not in the process working set or they are on secondary storage.

working set swapper:
A system process that brings process working sets into the balance set and removes them from the balance set.

world:
In the context "system, owner, group, world," world refers to all users, including the system operators, the system manager, and users both in an owner's group and in any other group.

write access type:
The specified operand of an instruction or procedure is written only during that instruction's or procedure's execution.

write allocate:
A cache management technique in which cache is allocated on a write miss as well as on the usual read miss.

write back:
A cache management technique in which data from a write operation to cache are copied into main memory only when the data in cache must be overwritten. This results in temporary inconsistencies between cache and main memory. Contrast with write through.

write through:
A cache management technique in which data from a write operation are copied in both cache and main memory. Cache and main memory data are always consistent. Contrast with write back.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

VAX Hardware Handbook
Volume 1–1986

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 _____

(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 Group
CFO 1-2/M92
200 Baker Avenue
West Concord, MA 01742

digital

Printed in U.S.A. EB 25949 46/87B 04 04/43 30.0