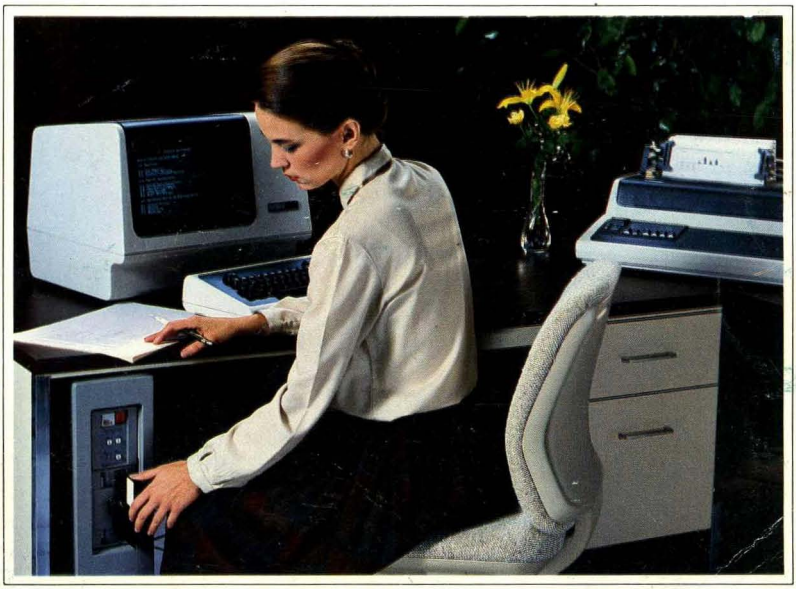
digital

PDP-11 MICRO/PDP-11 Handbook

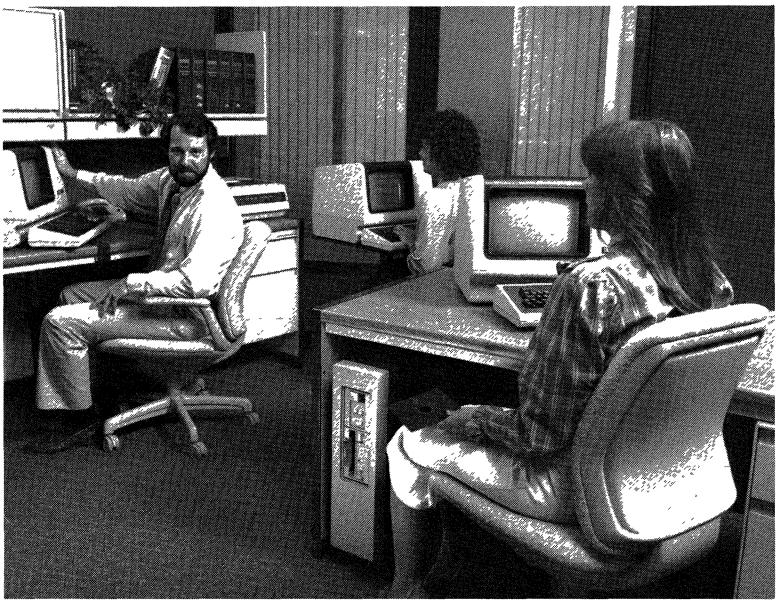
1983–84



PDP-11

MICRO/PDP-11 Handbook

digital



MICRO/PDP-11
The Team Computer

PDP-11

MICRO/PDP-11 Handbook

digital

Copyright © 1983 Digital Equipment Corporation
All Rights Reserved.

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

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

DATATRIEVE, DEC, DECdirect, DECnet, DECsystem-10, DECSYSTEM-20, DECtape, DECUS, DECwriter, DIBOL, Digital logo, IAS, MASSBUS, MICRO/PDP-11, MicroPower/Pascal, OMNIBUS, PDP, PDT, RSTS, RSX, SBI, UNIBUS, VAX, VMS, VT
are trademarks of
Digital Equipment Corporation

This book was produced on DIGITAL's DECset Integrated Publishing System. Book production was done by DIGITAL's Educational Services Development and Publishing in Bedford, MA.

CONTENTS

PREFACE

MICRO/PDP-11 is DIGITAL's Team Computer.

For only a bit more than some personal computers, MICRO/PDP-11 can make development, computation and communication resources available to members of your team at two to ten terminals. Proven multiuser operating systems allow programs and information to be shared among team members, when the originating member chooses. And you can draw on existing PDP-11 applications developed worldwide since 1970.

Here is the best of both worlds for people working as a team: the capability of a PDP-11 system, at a lower cost per terminal than many personal computers.

What if you're not working in a team?

MICRO/PDP-11 brings the real-time responsiveness of the PDP-11 architecture and event-driven operating systems to a new, flexible mechanical design that can be easily integrated into your instrumentation, control, or data acquisition application.

This handbook is your introduction to the MICRO/PDP-11. It offers you:

- Descriptions of the MICRO/PDP-11 system and related products: peripherals, interfaces, operating systems, communications software, and languages
- Help in identifying the MICRO/PDP-11 configuration that is right for your needs
- Answers to the most commonly asked technical questions, and direction to sources for additional information if you need it.

A companion volume to this handbook, the *PDP-11 Software Source Book,* can provide you with an annotated index to the enormous inventory of applications software that has been developed for the PDP-11 since 1970.

CHAPTER 1
INTRODUCTION

MICRO/PDP-11

If you are looking for an inexpensive multiuser computer system, there are many to choose from. They all use a microprocessor, and they all use small Winchester disks. They even seem to look the same.

DIGITAL's MICRO/PDP-11 is the one that stands out. Like the others, it uses a microprocessor, but its microprocessor is a full-fledged PDP-11—a real difference. Why does this make a difference?

- People have been writing programs for the PDP-11 since 1970. When you use a MICRO/PDP-11, you can take advantage of the software—operating systems, languages, tools and applications—developed by DIGITAL, our customers, and independent companies.
- The LSI-11 bus has been a part of DIGITAL systems since 1975. When you use the MICRO/PDP-11, you can fit the system to your application with LSI-11 bus interfaces and peripherals designed by DIGITAL, our customers, and independent manufacturers.
- As your needs grow, MICRO/PDP-11 gives you three growth paths: VAX-11 systems with their similar architecture, language compatibility and common file structures, larger PDP-11 systems, and planned future enhancements to the MICRO/PDP-11.
- MICRO/PDP-11 does not stand alone. It can exchange data with DIGITAL's personal computers, participate in a public patch-switched network, join a DECnet network, and communicate with other manufacturers' systems.

You will find the MICRO/PDP-11's size and flexibility convenient. Imagine a tabletop system—6 inches high, 21.5 inches wide, 25 inches deep—with a PDP-11 processor, up to 2MB of memory, a

10MB Winchester disk, dual 400KB diskettes, two serial-line connectors, and LSI-11 bus slots for additional interfaces or expansion (Figure 1-1).

Now picture the same system rotated 90 degrees and mounted under a desk or table (Figure 1-2).

Does your application require rack-mounted equipment? MICRO/PDP-11 is also available in a 5.2-inch-high rack-mount version (Figure 1-3).

MICRO/PDP-11 is the low-cost system that lets you put a real PDP-11 where you never could before. That is the difference.

MICRO/PDP-11 SOFTWARE AND SERVICE

The hardware is the part of the system you see, but software and service—the parts of a system you don't see—are equally important.

Software

System software doesn't appear overnight. It takes time to develop and test. It takes even more time when every effort is made to maintain compatibility in order to continue use of existing software and to ensure a long, useful life for newly developed software. DIGITAL system software has been proven in the field. Each operating system or language is regularly updated to support new devices, to extend the features available to the user, to increase its reliability and ease of use, and to facilitate service. In all these efforts compatibility is emphasized—in user interface, in file structure, in utilities, and in languages—wherever practical.

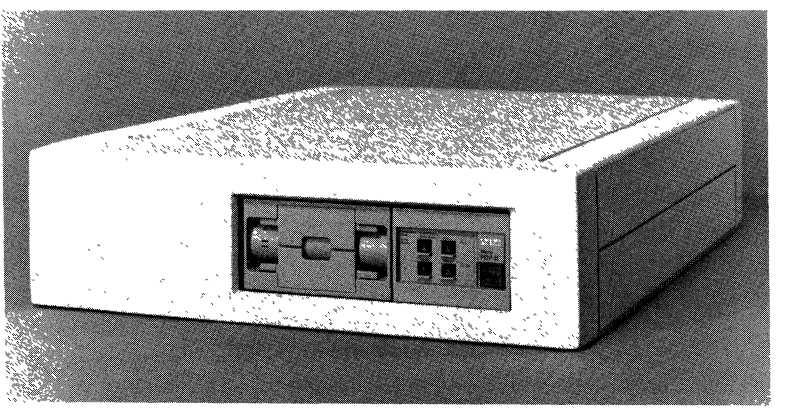


Figure 1-1 Tabletop MICRO/PDP-11

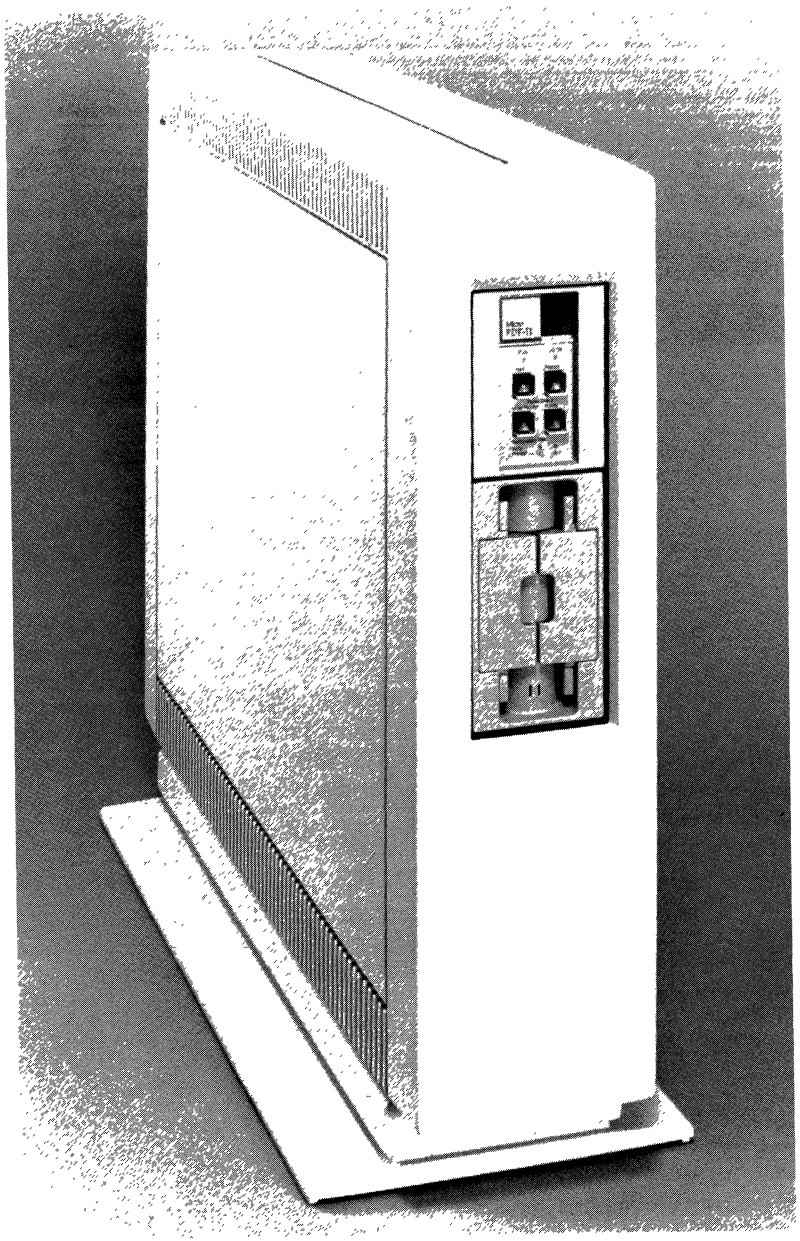


Figure 1-2 Floor-mount MICRO/PDP-11

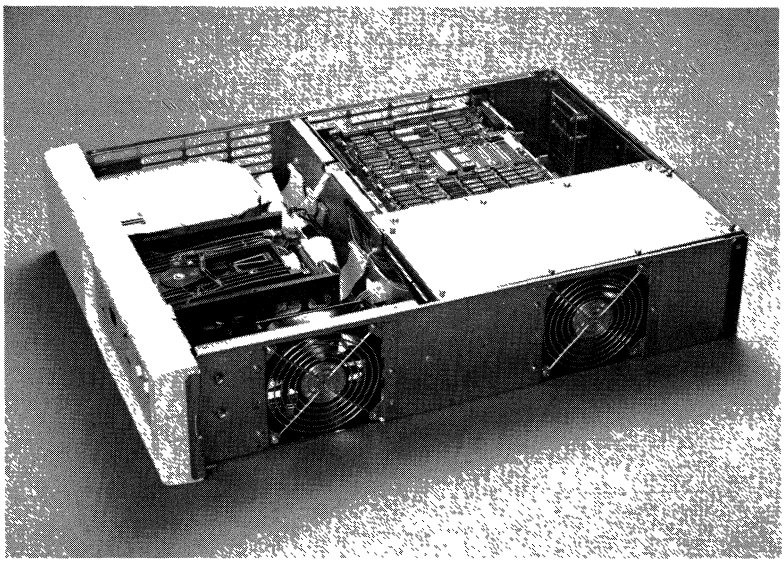


Figure 1-3 Rack-mount MICRO/PDP-11

Operating Systems—The operating system allocates system resources, manages the file system, takes care of the system housekeeping functions, and protects one user's programs and files from being disrupted by other users (while protecting the system software from all users). With a MICRO/PDP-11, you can choose the operating system with the set of attributes you need:

- RSX-11M and RSX-11M-PLUS are multiprogramming, multiuser operating systems optimized for fast response to real-time events. Individual system users have unique IDs, passwords, file directories, and access privileges. RSX-11M-PLUS is a compatible superset of RSX-11M for larger memory systems. RSX-11S is a memory-resident subset for run-time support of application programs.
- RSTS/E and Micro/RSTS provide interactive timesharing for applications, such as small businesses and educational institutions, where the emphasis is on ease of use and quick response to human users at multiple terminals. Micro/RSTS is a run-time subset ideal for maximizing user storage when executing applications on the MICRO/PDP-11.
- RT-11 is a single-user system, optimized for fast, low-overhead response to real-time events. Concurrent program execution

and program development are supported in foreground/background mode. CTS-300 adds multiterminal DIBOL for commercial applications.

- MicroPower/Pascal includes a high-performance optimizing Pascal compiler and software tools for developing real-time application programs on a host machine, and a run-time software environment to execute these applications on a target machine.
- DSM (DIGITAL Standard MUMPS) is a timeshared operating system combined with the MUMPS programming language. Originally developed at the Massachusetts General Hospital for medical applications, MUMPS' data base orientation and ease of use have made it attractive for commercial applications as well.
- UNIX V7M-11. Since Bell Laboratories created PDP-11 UNIX in 1971, users and programmers have probably had more UNIX experience on PDP-11 systems than on any other architecture. UNIX is a multiprogramming environment with a hierarchical file system and compatible file, device, and interprocess I/O. The structured high-level languages C and RATFOR are included, along with FORTRAN-77.

High-Level Languages—Whereas the operating system manages the system resources, you may want to write a program in one of the high-level languages available on the MICRO/PDP-11. In addition to the languages described below, UNIX, DSM, and MicroPower/Pascal include high-level languages as part of the operating system.

- Either of the MICRO/PDP-11's two implementations of FORTRAN is a good choice for computation, real-time control, or general data processing applications. FORTRAN IV is a fast, one-pass compiler providing a superset of the 1966 ANSI standard. FORTRAN-77 is an extended implementation of the 1978 ANSI standard, producing optimized code and taking full advantage of optional hardware floating-point support.
- MICRO/PDP-11 users can choose among three versions of BASIC—the easy-to-learn conversational language, suitable for commercial, technical, and educational applications. BASIC-11 and BASIC-PLUS are interactive, with BASIC-PLUS (included in RSTS/E systems) offering additional string operators for commercial applications. BASIC-PLUS-2 is a compiled superset of BASIC-PLUS, supporting full file and data management capability.

- COBOL-81 emphasizes file processing and data movement, as found in business applications. It complies with the ANSI 1974 specification and makes full use of optional hardware character and decimal string support.

Network Architecture—DIGITAL's Network Architecture (DNA) defines an overall approach to networking; a set of software and hardware products support a range of requirements.

- MICRO/PDP-11 and other DIGITAL computers use DECnet to share files, programs, and resources. Systems can communicate over traditional interconnects as well as Ethernet for high-speed local communication.
- By emulating the protocol of other manufacturers' devices, MICRO/PDP-11 products with Internet software can communicate with other vendors' equipment. IBM batch (2780/3780), interactive (3271), and SNA protocols are supported along with CDC and UNIVAC.
- A MICRO/PDP-11 with DIGITAL's Packetnet system interface can communicate through a public packet-switched network (X.25) with other systems regardless of manufacturer.

Some software tools and applications available from DIGITAL:

- FMS-11 can handle all the complexity of forms-oriented video data management, so you can concentrate on your application.
- DATATRIEVE gives you interactive access to files, allowing you to ask questions, write reports, and manage the data.
- If you are not a programmer, ADE helps you interactively develop software for character, numeric, or date-oriented applications.

A great deal of software has been developed for PDP-11 systems by producers of software other than DIGITAL. For more information on programs that might be applicable to your requirements, see the *PDP-11 Software Source Book*.

Service

Like all DIGITAL products, the MICRO/PDP-11 has been designed for reliability and manufactured to strict quality control standards that ensure each unit meets its design goals.

DIGITAL's respected service organization is ready to follow up with quality support if you need it.

You may be able to install your MICRO/PDP-11 system yourself (depending on the hardware and software options you buy with it) and then run its easy-to-use system diagnostics to ensure that it is func-

tioning correctly. If you have a question about installation or operation, there is a telephone help line you can call to get an answer.

If problems do arise, you have three choices for hardware service:

- DIGITAL's trained service personnel are prepared to service your system at your location, either under a service contract or on a time-and-materials basis.
- If you maintain your own MICRO/PDP-11, you can take a faulty component (field-replaceable unit) to one of DIGITAL's carry-in service centers for repair.
- Or you can mail a faulty module to DIGITAL for repair under the DECmailer program.

DIGITAL's Software Services personnel are available for installation and maintenance for support if you maintain DIGITAL software yourself. Courses are regularly offered on both the hardware and software aspects of the system.

MICRO/PDP-11 IN THE DIGITAL PRODUCT FAMILY

MICRO/PDP-11 is part of a family of microprocessor products. Systems, board-level processors, single-board computers, and microprocessor chips are available to meet your requirements. All draw on the architectural heritage of the PDP-11 and use the widely accepted LSI-11 bus for easy system expandability and compatibility with custom or third-party devices. All support one or more of DIGITAL's proven 16-bit operating systems for program development, multiuser timesharing, multistream batch, or real-time processing such as data acquisition or process control.

Microprocessor Chips

Following are descriptions of current microprocessor chips:

- MICRO/J-11 is DIGITAL's top-of-the-line microprocessor. The full PDP-11 architecture—dual register sets, three processor operating modes, instruction and data space, 4MB physical addressing, all data types—is implemented with performance comparable to the powerful PDP-11/70 supermini (Figure 1-4).
- MICRO/F-11, DIGITAL's mid-range microprocessor, has all the PDP-11 architectural elements needed for an efficient multiprogramming system environment, with performance comparable to traditional minicomputers such as the PDP-11/34 (Figure 1-5).
- MICRO/T-11 packs the essential elements of the PDP-11 architecture in a single, low cost, 40-pin package that interfaces easily to industry standard peripheral chips.

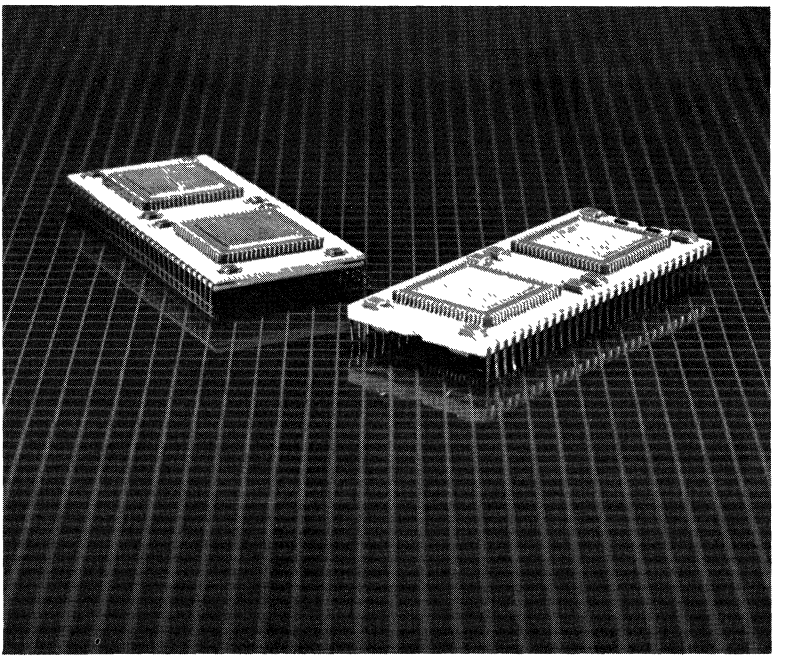


Figure 1-4 J-11

Processor Modules

Available processor modules include the following:

- FALCON (KXT11-A) uses the MICRO/T-11 microprocessor, along with PROM for software storage, memory, and I/O—all on a single 5.25-inch by 8.5-inch board.Expansion is easy since FALCON can plug in to the LSI-11 bus (Figure 1-6).
- The KDF11-B LSI-11 bus processor module is a quad-height module using an F-11 microprocessor. Included are on-board self-test diagnostics, bootstrap microcode, serial lines for the console, and an additional terminal (Figure 1-7). Floating-point instructions or character- and decimal-string instructions are optional.
- The KDF11-A LSI-11 bus processor module is a double-height module using the F-11 microprocessor. Floating-point instructions are optional.

MICRO/PDP-11 takes its place in the DIGITAL family of system products along with VAX-11, the Professional 300 personal computers,

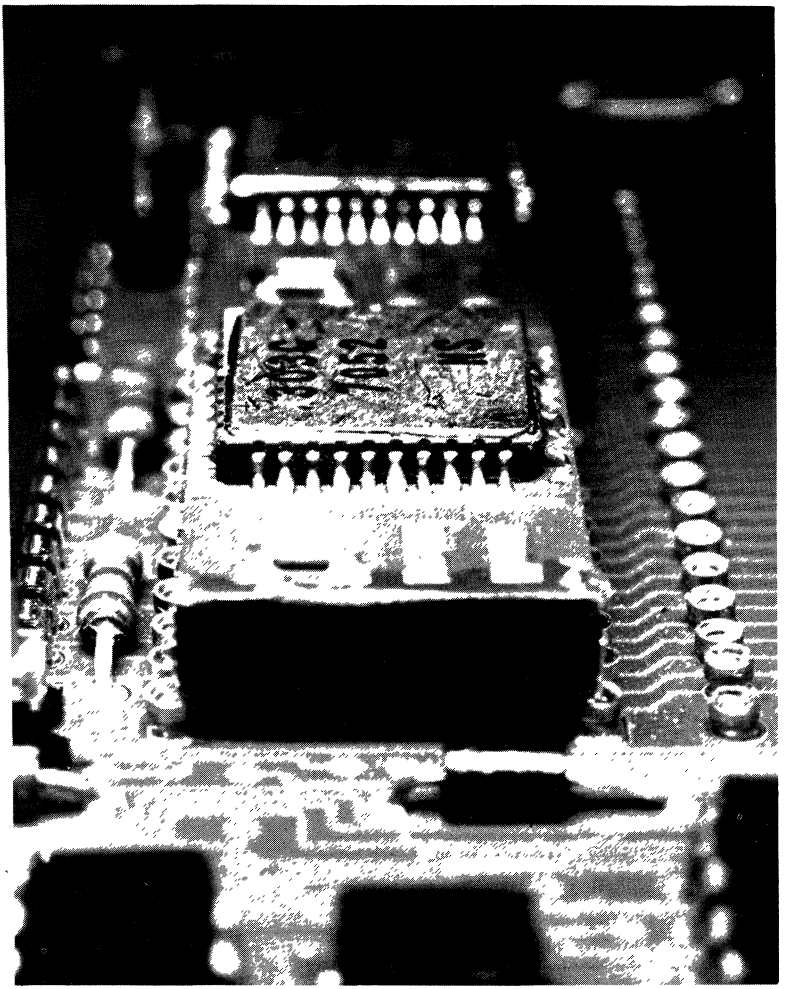


Figure 1-5 F-11

and the UNIBUS PDP-11 systems. The following comparisons may help you narrow your decision.

MICRO/PDP-11 or VAX-11?

You may prefer a MICRO/PDP-11 if:

- Low cost is a key factor in your decision
- Real-time responsiveness is important

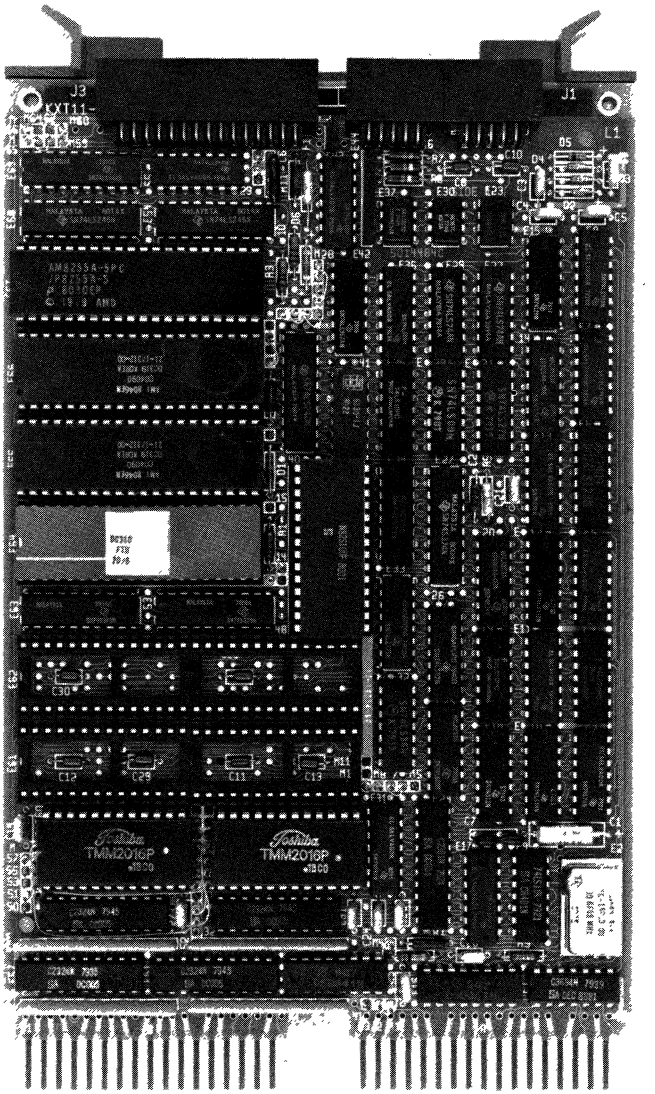


Figure 1-6 FALCON

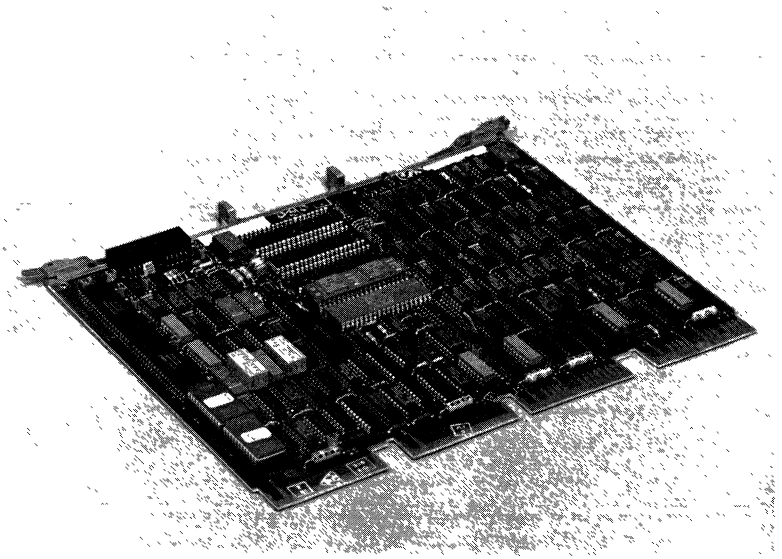


Figure 1-7 KDF11-B

- You prefer under-desk or tabletop mounting
- You have (or need) existing PDP-11 software applications
- Software compatibility with Professional 300 is important.

You may prefer a VAX-11 if:

- You want software products available on VAX but not on PDP-11
- You need the processing power of a superminicomputer like the VAX-11/780
- You need to support more than 10 users simultaneously
- Your applications and programs are large
- You have a new application and anticipate significant future growth.

MICRO/PDP-11 or UNIBUS PDP-11?

You may prefer a MICRO/PDP-11 if:

- Low cost is a key factor in your decision
- You prefer under-desk or tabletop mounting
- You have a new application
- You have previously used LSI-11 Bus (Qbus) products.

You may prefer a UNIBUS PDP-11 if:

- You have an investment in UNIBUS PDP-11 products
- Your application requires larger mass storage
- You need higher performance now.

MICRO/PDP-11 or Professional 300?

You may prefer a MICRO/PDP-11 if:

- Lowest cost per user for multiple users is a key factor in your decision
- Your application requires real-time interfacing and control
- Files, data bases, or computation are important in your application
- You need DECnet, Internet, or Packetnet communications nodes
- Software products you want are available on MICRO/PDP-11 but not on the Professional 300.

You may prefer a Professional 300 if:

- Lowest cost for a single user is a key factor in your decision
- Your application calls for character-intensive work, high-performance graphics, or telephone management
- Software products you want are available on the Professional but not on MICRO/PDP-11.

Start Anywhere

No matter which of these systems you start with, you can grow to use all of them together. The architectures, including data formats and instructions, are similar or identical. The user interfaces, through high-level languages or through the operating system command language, are often compatible. Many of the file structures and on-disk structures are compatible, facilitating data interchange. And the DIGITAL Network Architecture provides the means for interaction among them all.

WHO IS DIGITAL?

Digital Equipment Corporation was founded in 1957.

In 1957, a computer was something kept in a big, air-conditioned room. Most users saw it only through a plate-glass window: row after row of cabinets, some with dozens of lights, others with spinning reels of tape. The computer was awesome and unapproachable.

That was all right because nobody wanted you, the user, to approach the computer. You punched your program into a stack of cards and

turned them over to the system operator. With any luck, you could expect to get your output late that day or early the next.
Users could wait for the computer because in 1957 the computer was too expensive to wait for the users.

DIGITAL began to redefine the computer with the PDP-1, first delivered in 1960 (Figure 1-8). It was much smaller, and you could sit right down at its typewriter, video display, or console and interact with the computer.

In 1965, DIGITAL shipped the first PDP-8, and the redefinition of the computer took another big step (Figure 1-9). The first minicomputer found its way into labs, schools, and industry . Other companies—the original equipment manufacturers (OEMs)—began to build the PDP-8 into larger systems they designed for specific applications.

DIGITAL's PDP-8 started the minicomputer industry.

In 1970, DIGITAL introduced the PDP-11, a new computer architecture taking full advantage of technology and the company's experience

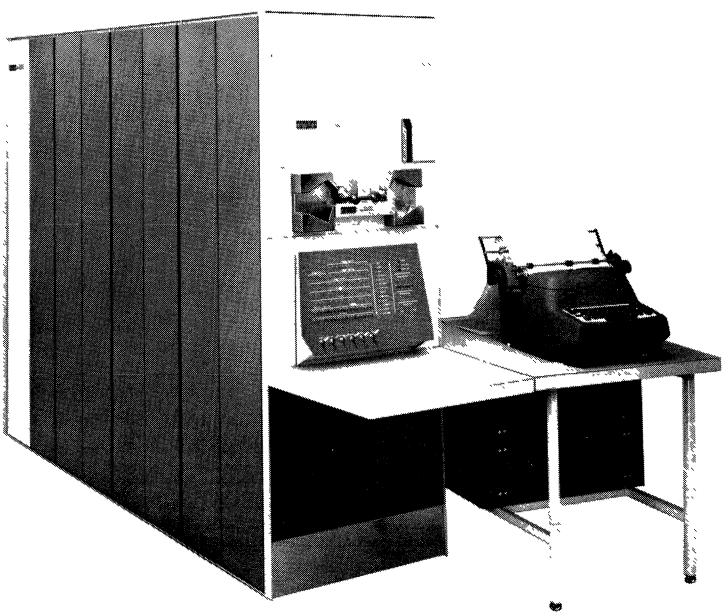


Figure 1-8 PDP-1

with the PDP-8. Since 1970, the PDP-11 architecture has been delivered in machines ranging from a low-cost single-board computer to 64-user timesharing machines costing several hundred thousand dollars.

DIGITAL's PDP-11 established the standard of the minicomputer industry.

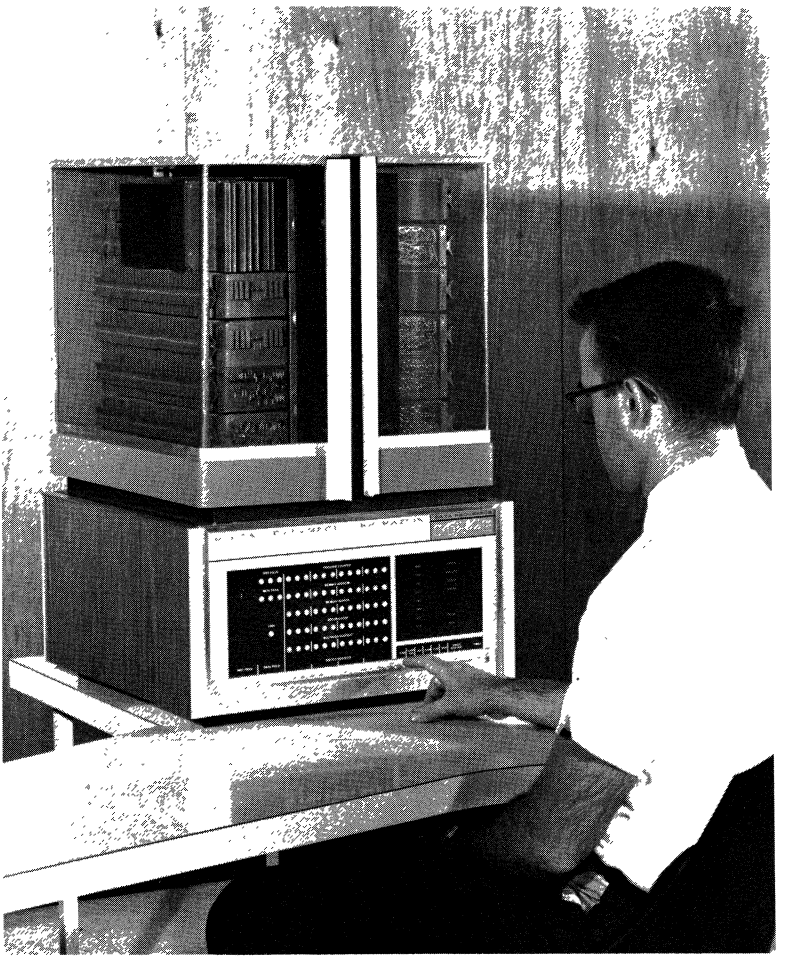


Figure 1-9 PDP-8

In 1978, continuing technological development and experience with the PDP-11 family led DIGITAL to introduce the VAX-11 (Figure 1-10). This new architecture gives users more system capability while retaining a high level of compatibility with PDP-11 programs and data.

DIGITAL's VAX-11 established the standard of the supermini market.

In 1982, DIGITAL announced a family of personal computers (Figure 1-11). The Professional 300 uses an integrated circuit PDP-11, and DECmate II uses an integrated circuit PDP-8. People using these computers have access to the experience and software developed on DIGITAL minicomputers since 1965.

DIGITAL is the computer company that has made computers affordable, approachable, and available. The success of these computer systems has made DIGITAL the second-largest company in the industry.

THE MICRO/PDP-11'S FAMILY TREE

The first member of the PDP-11 family, the PDP-11/20, was shipped in 1970. Since then, over a dozen PDP-11 models have been introduced and the architecture has been expanded.

The PDP-11 *architecture* refers to the major characteristics of the system as seen by the programmer: memory size, addressing scheme, number of registers, data types, and instructions.

Each model represents a different *implementation* of the PDP-11 architecture; technology, interconnects, cost, performance, and architectural completeness vary from model to model.

New PDP-11 models have progressed in three directions from their predecessors:

- *Maximum performance* systems cost more than their immediate predecessors. These systems have additional functions and the highest performance reasonably achievable with current technology.
- PDP-11s developed with *optimum cost/performance* as their goal cost about the same as their immediate predecessors. They use technological advances to provide additional functions and better performance at that price.
- *Low-cost* designs have offered functions and performance similar to their predecessors at a lower cost due to improvements in technology.

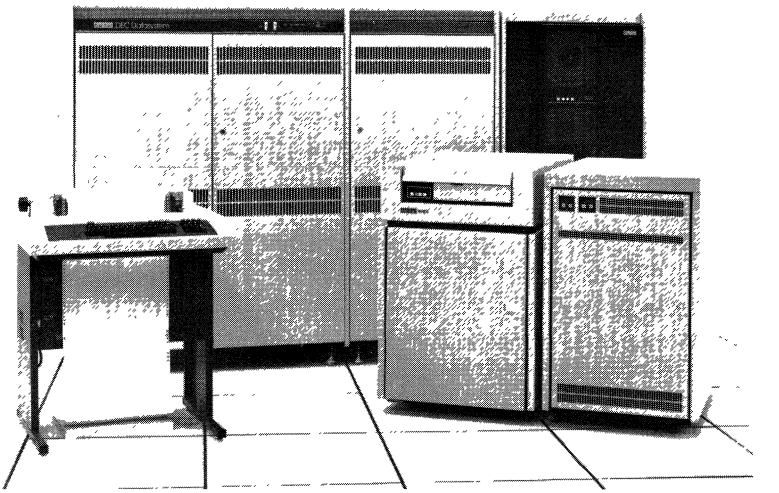


Figure 1-10 VAX-11

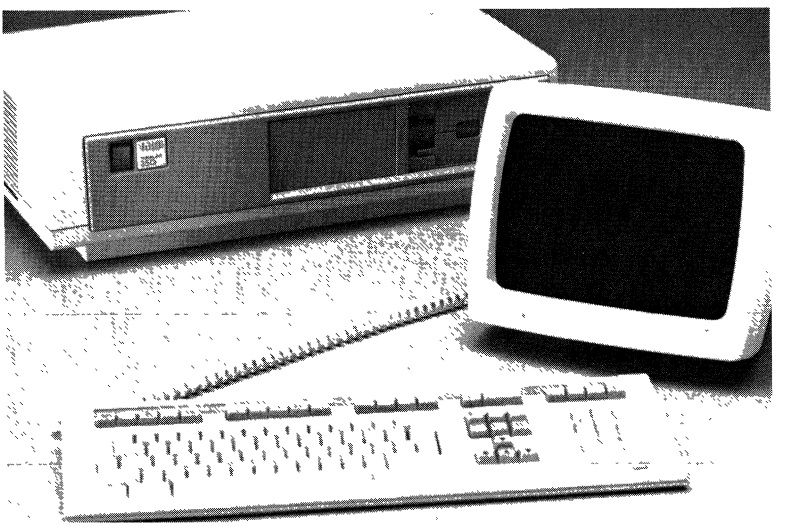


Figure 1-11 DIGITAL's Professional 350 Personal Computer

1970

Architecture—The PDP-11/20 introduced the PDP-11 family to the world. Some of the attributes the world noticed included:

- Addressing. Memory, processor registers, and I/O devices are alladdressed using the same instructions. The programmer can choose addressing modes to use the PDP-11 as a stack machine, a general-register machine, or as a two-address machine.
- Character data. Direct byte addressing simplifies character handling.
- Registers. Instead of a single accumulator, the PDP-11/20 offered eight general registers.
- Interrupt handling. PDP-11 interrupt vectors rapidly switch processor context to begin executing the interrupt routine.
- I/O. High performance devices can transfer to or from memory without processor intervention using direct memory access (DMA).

Implementation—The PDP-11/20 CPU used small-scale integrated circuits, core memory, and a hard-wired processor. It introduced the UNIBUS as a PDP-11 interconnect.

1972

Architecture—With the introduction of the PDP-11/45, new capability was added to the PDP-11 architecture:

- Memory management. The 64KB physical address space (memory capacity) of the PDP-11/20 was expanded to 256KB, and three processor operating modes were specified for protection in multiprogrammed systems.
- Floating-point data. New instructions, along with six floating-point registers, supported faster floating-point arithmetic.
- Integer arithmetic. New instructions were added for better multiply, divide, and shift performance.

Implementation—The PDP-11/45 CPU was implemented with faster medium-scale integrated circuits and a microprogrammed processor. A fast memory interconnect, not accessible to the user, worked with the UNIBUS to improve performance. PDP-11/45 variations were available with core memory, as on the PDP-11/20, with MOS semiconductor memory or with fast bipolar semiconductor memory. Floating-point instructions were executed by a fast, autonomous floating-point processor.

1975

Architecture—The PDP-11/70 became the first PDP-11 with a physical address space of 4MB.

Implementation—The PDP-11/70 CPU is the highest performance PDP-11 (Figure 1-12). It was implemented with a fast, 2KB bipolar cache memory. High-speed interconnects were used between the cache and main memory and between the cache and high-performance DMA peripherals, in addition to the UNIBUS.

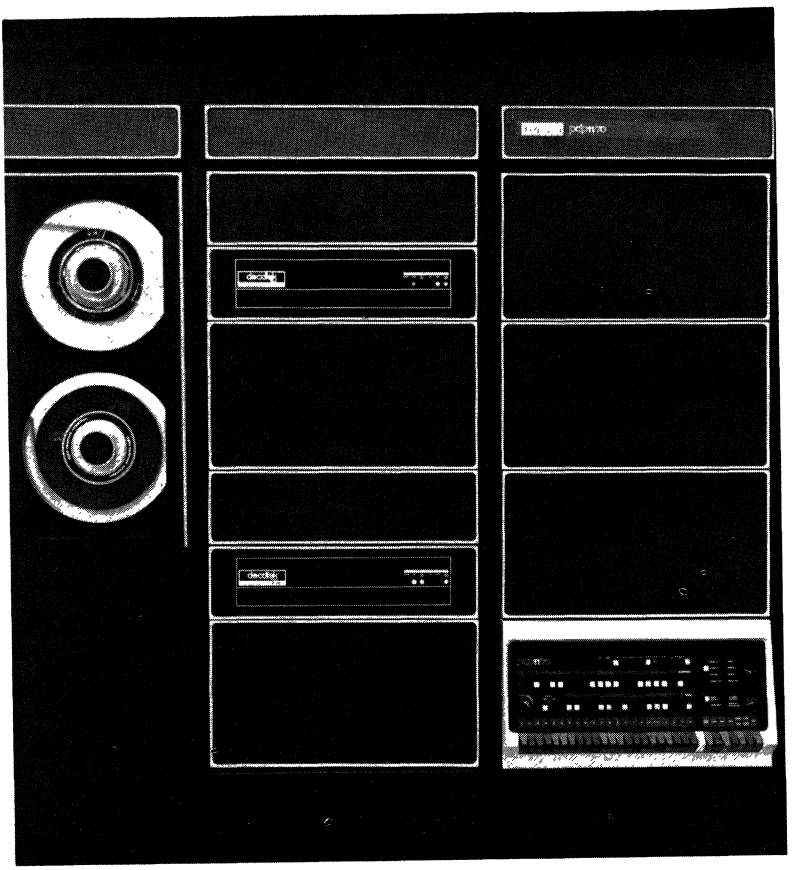


Figure 1-12 PDP-11/70

The PDP-11/03 used LSI semiconductor technology to build the PDP-11 architecture at a low price (Figure 1-13). All interconnects among the processor, memory, and peripherals used the new LSI-11 bus, similar in concept to the UNIBUS but with fewer lines for lower cost. Memory capacity was limited to 64KB, only a single processor operating mode was implemented, and a limited set of floating-point instructions was supported.

1979

Architecture—The PDP-11/44 added character-string and decimal-string data types, with new instructions for their use, to the PDP-11 family architecture (Figure 1-14).

Implementation—The PDP-11/23 became the second LSI-11 bus processor, taking advantage of advances in LSI technology to provide more of the PDP-11 architecture and better performance at PDP-11/03 cost (Figure 1-15). Memory capacity was increased to 256KB, a second processor operating mode was added, and the complete set of floating point instructions was supported.

1981

Implementation—The PDP-11/23-PLUS brought another element of the PDP-11 architecture to low-cost systems by increasing the physical address space to 4MB and implementing character- and decimal-string instructions.

1983

Implementation—The MICRO/J-11 microprocessor makes available to users of low-cost systems the full PDP-11 architecture together with the performance of the PDP-11/70.

MICRO/PDP-11 brings multiuser PDP-11 systems to the lowest price yet in flexible, convenient system packages . Access to the PDP-11 system and application software base, together with the proven and expanding set of LSI-11 bus peripherals and interfaces, delivers unparalleled price/ performance.

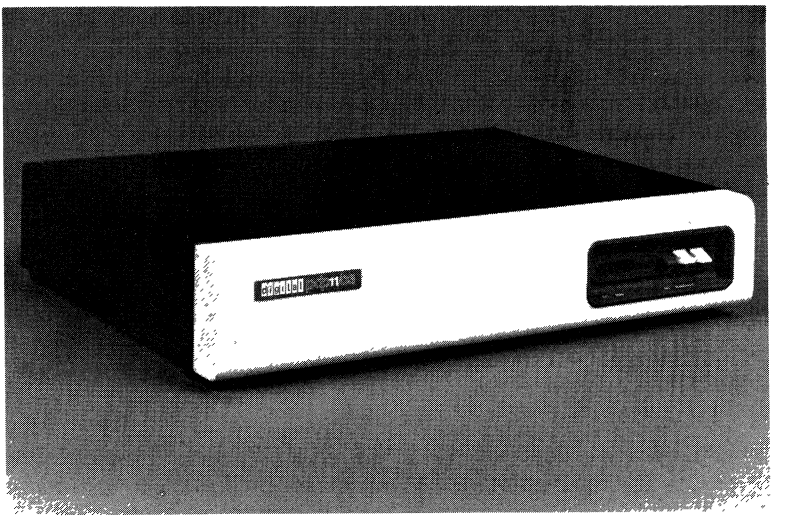


Figure 1-13 PDP-11/03

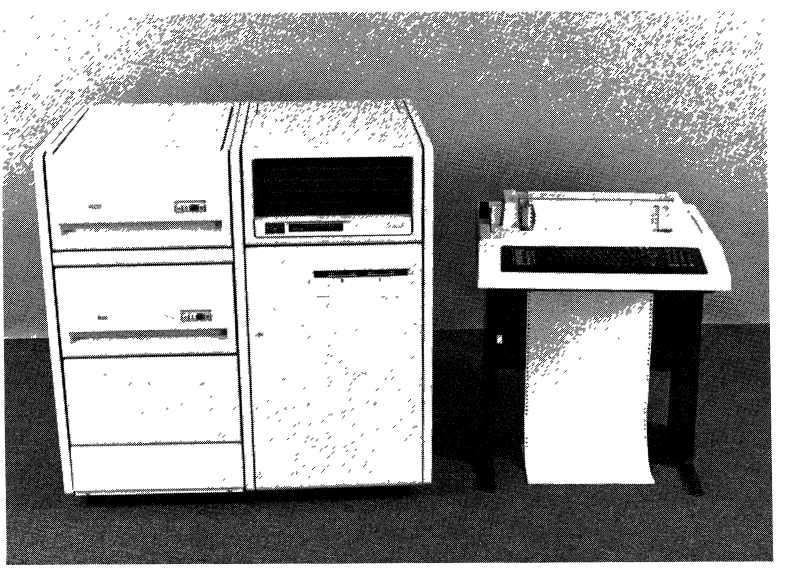


Figure 1-14 PDP-11/44

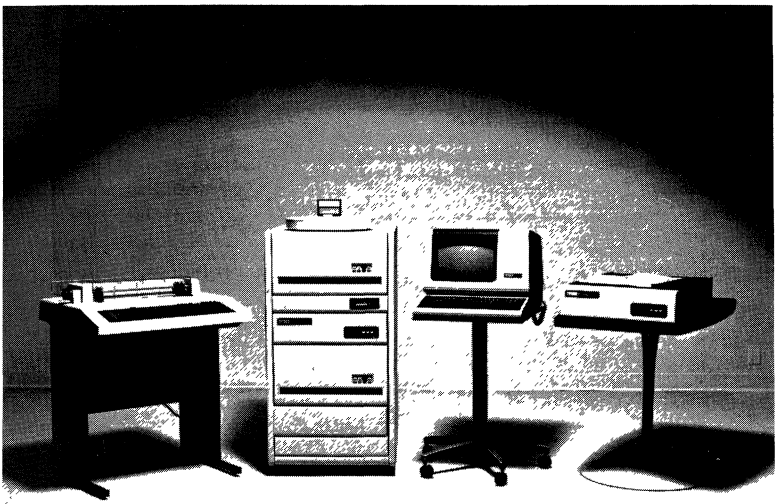
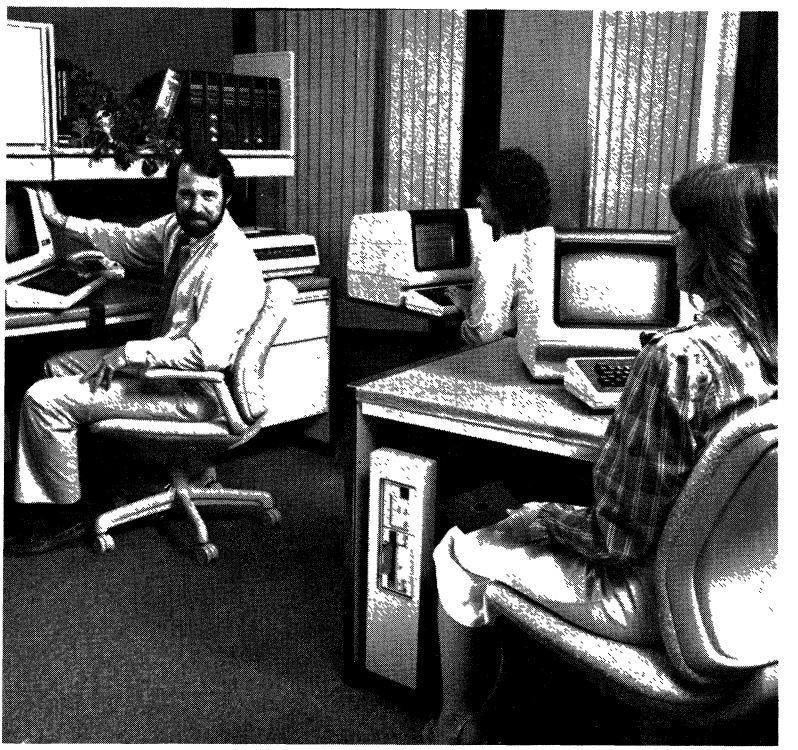


Figure 1-15 PDP-11/23

The Future

Implementation—MICRO/PDP-11 systems are designed with growth in mind. The MICRO/J-11 microprocessor, as part of a future MICRO/PDP-11 system, will make possible the full PDP-11 architecture and high-end PDP-11 performance at the low price of a MICRO/PDP-11.

As memory technology advances, 256K chips will be used in new memory modules with more capacity per board and other functional enhancements. New mass-storage devices with more capacity will be added. And new LSI-11 bus interfaces are in the works.



CHAPTER 2
MICRO/PDP-11 SYSTEMS

The compact size of the MICRO/PDP-11 gives new flexibility to the appearance, location, and mounting of multiuser PDP-11 systems. A complete MICRO/PDP-11 system, with 10MB Winchester disk and dual 400KB diskettes, can sit on a tabletop, stand on the floor under a desk or table, or insert in a standard 19-inch rack.

When comparing computer systems, it's tempting to simplify the comparison by focusing on the common denominator—the CPU. Since computer systems process data, however, it is important to be able to move data into and out of the system and to store data in the system. MICRO/PDP-11 inherits a broad range of LSI-11 bus options (described below) that address most storage and I/O applications. And, since DIGITAL's LSI-11 bus systems and components have been popular since 1975, storage and I/O devices are also available from other manufacturers for more specific applications.

MICRO/PDP-11 storage and I/O options rules are covered in more detail in Appendix C. Configuration guidelines are in Appendix B.

The MICRO/PDP-11 is a flexible, highly functional system today. DIGITAL is committed to future product enhancements:

- More power with the MICRO/J-11 microprocessor
- New memory technology with 256K RAM
- Larger capacity Winchester disk storage
- Additional LSI-11 bus options.

MICRO/PDP-11 SYSTEM CHASSIS

The system chassis is the common denominator of the MICRO/PDP-11 family. The major components (Figure 2-1) are listed below:

- RD51 10MB Winchester disk
- RX50 dual 400KB diskette drive

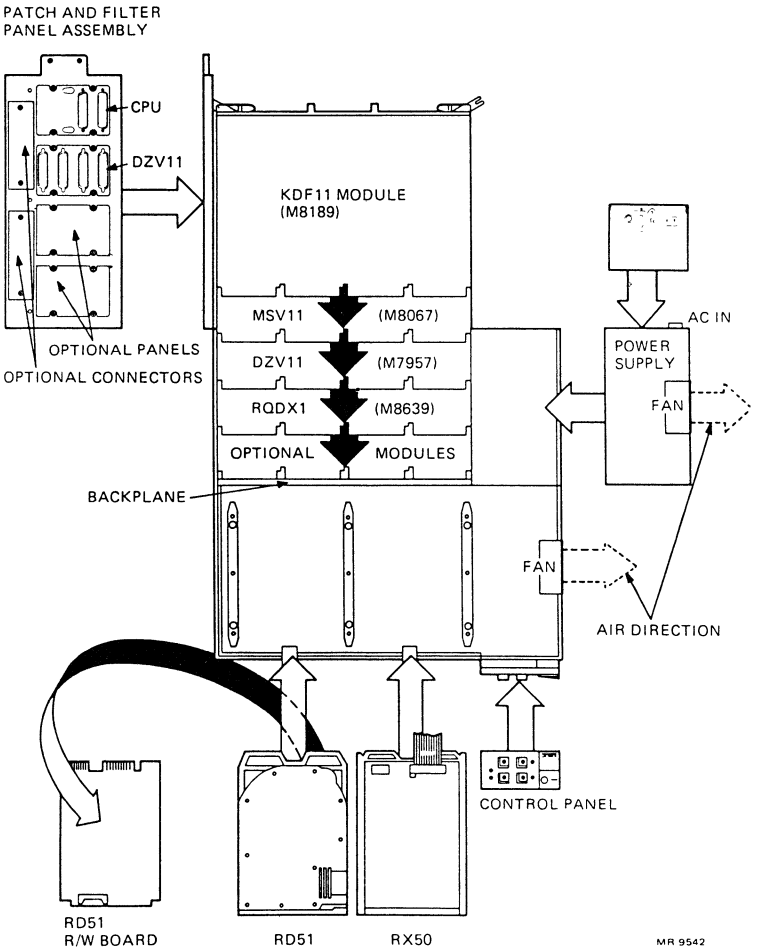


Figure 2-1 Exploded View of System Chassis

- Logic assembly, including:
  - KDF11-B CPU module
  - MSV11-P memory, 256KB or 512KB
  - DZV11 four-line multiplexer
  - RQDX1 controller module for RD51 and RX50
  - Backplane and card cage
  - Expansion slots for eight dual options or four quad options

- Power supply
- System control panel
- System I/O connection panel.

KDF11-B CPU Module

The KDF11-B multifunction CPU module provides all PDP-11 architectural elements needed for a multiprogramming system:

- 4MB physical address space
- Integer data types and instructions (standard)
- Floating-point data types and instructions (optional)
- Character- and decimal-string data types and instructions (optional)
- Joint instruction and data space, not separate
- Kernel and user operating modes only, no supervisor mode.

Future CPU modules based on the MICRO/J-11 microprocessor will include supervisor mode and separate instruction and data spaces, to provide the complete PDP-11 architecture.

In addition to the basic CPU, the KDF11-B has:

- Two asynchronous serial lines, with externally selectable baud rates
- ROM-based diagnostic code to self-test the CPU, memory, and console terminal on power-up or restart
- ROM-based bootstrap code to load software from a designated storage device or DECnet line on power-up or restart
- Line frequency clock
- Module status LED display to help diagnose system faults.

MSV11-P Memory

MSV11-P memory provides 256KB or 512KB of parity memory per module, using 64K RAM technology. An LED status display on each module helps in diagnosing system faults. MSV11-P works with the RQDX1 storage controller to transfer data with efficient block-mode direct memory access (refer to Chapter 7 for more on block-mode DMA.)

DZV11 Multiplexer

The DZV11 multiplexer interfaces to four asynchronous serial lines with auto-answer modem control. A 64-character first-in-first-out buffer for input characters prevents data loss. Because all four lines interface to the bus through a single controller, the software overhead is reduced.

RD51 10MB Disk, RX50 Diskette, and RQDX1 Controller

The RD51 Winchester disk drive uses two nonremovable 5.25-inch diameter platters as storage media, with one head for each of the four recording surfaces (Figure 2-2). The formatted capacity is 10MB.

The RX50 diskette drive is a single mechanism that accepts two single-sided 5.25-inch diameter diskettes, each with a capacity of 400KB (Figure 2-3). Each diskette has a separate access door and slot, and a "drive active" light.

A MICRO/PDP-11 system with either the RD51 or the RX50, or both, communicates with them through the RQDX1 controller. RQDX1 interfaces to the bus with block-mode direct memory access for maximum throughput and to the RD51 and RX50 with DIGITAL's mass storage control protocol (MSCP).

The MICRO/PDP-11 chassis can accommodate up to two RD51 Winchester disk drives, or one RD51 and one RX50 dual diskette drive, in the system chassis.

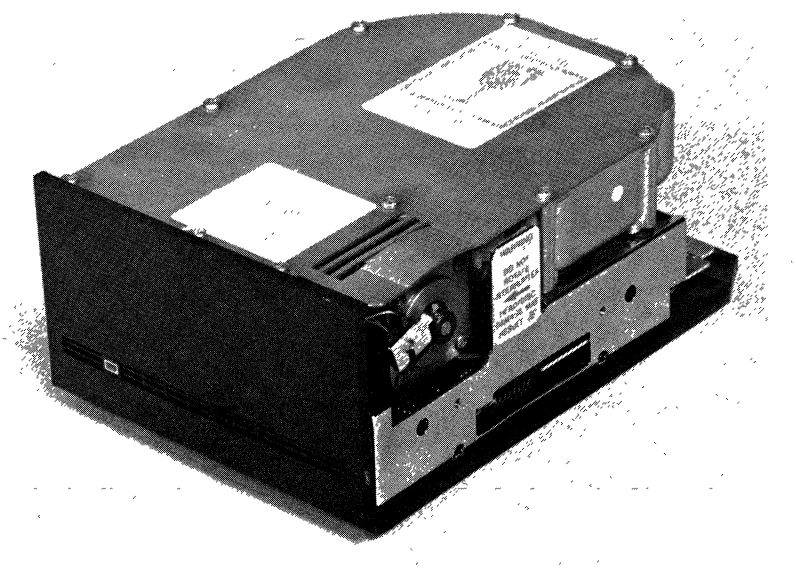


Figure 2-2 RD51

Logic Backplane

The MICRO/PDP-11 backplane is designed to support combinations of double and quad LSI-11 bus modules (Figure 2-4). There are eight quad slots in all: Slots 4–8 can accommodate two double modules or one quad, and Slots 1–3 can accommodate one module each, double or quad size.

In MICRO/PDP-11 systems, the CPU mounts in Slot 1, with memory in Slot 2 (and in subsequent slots if additional memory is used). The interrupt request and DMA request lines follow the pattern shown in Figure 2-5. The closer a module is to the CPU along this pattern, the higher its priority will be.

The sections of Slots 1–3 designated "C-D" are not available for mounting a double module.

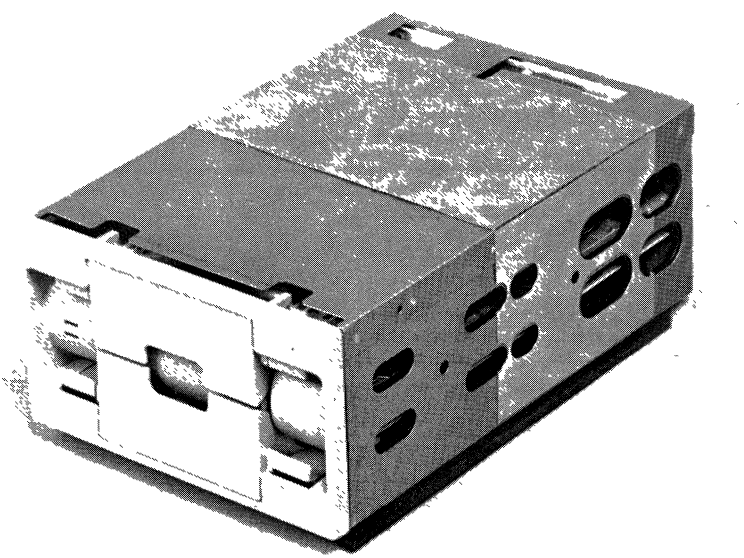


Figure 2-3 RX50

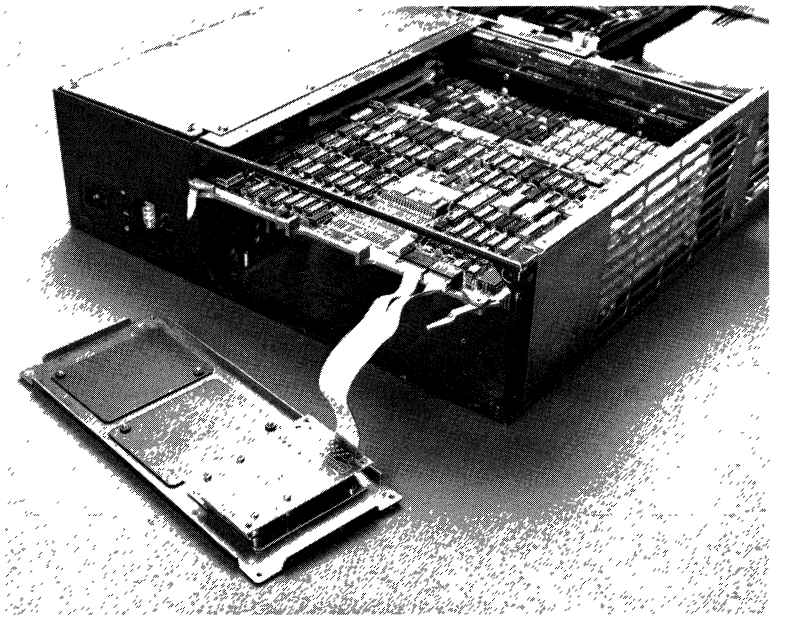


Figure 2-4 MICRO/PDP-11 Backplane/Card Cage

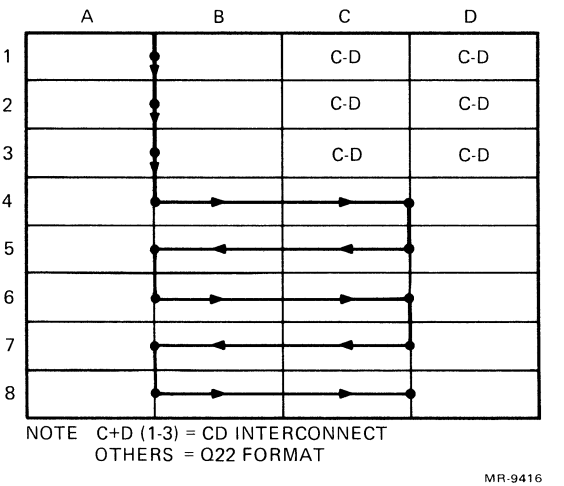


Figure 2-5 MICRO/PDP-11 Bus Pattern on Backplane

H7864 Power Supply

The power supply provides 230 watts of +5 V and +12 V for the logic modules and mass storage devices (RD51, RX50) mounted in the MICRO/PDP-11 chassis. The power supply also powers the system cooling fans.

System Control Panel

From the front panel of the MICRO/PDP-11, you can control system power, halt or restart the CPU, write-protect the fixed disk, and monitor the status of the CPU and mass storage devices (Figure 2-6).

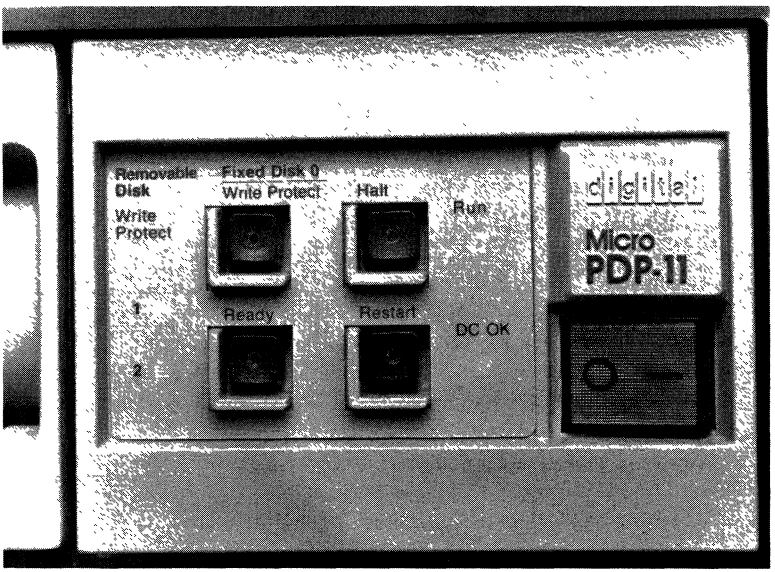
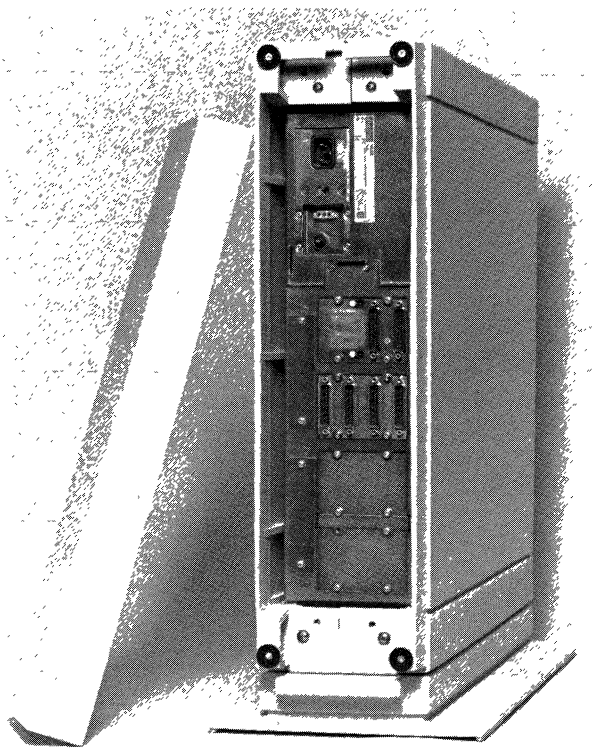


Figure 2-6 MICRO/PDP-11 Front Panel

System I/O Panel

All interconnections to the MICRO/PDP-11 are made through a system I/O panel at the back of the chassis (see Figure 2-7). The system I/O panel provides an accessible location for cable attachment, as well as for filtering and effective shield grounding necessary to keep electromagnetic and radio frequency interference (EMI/RFI) low.

Each module that sends a cable outside the MICRO/PDP-11 chassis has an insert that mounts in the system I/O panel; the external cable



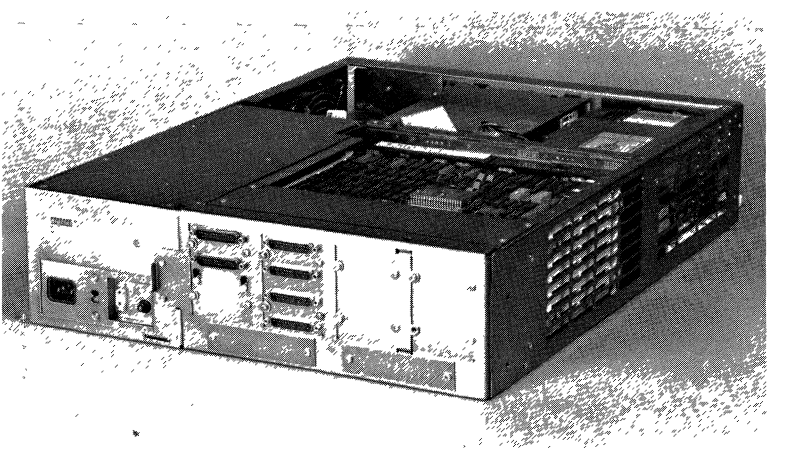


Figure 2-7 MICRO/PDP-11 Distribution Panel

attaches to a connector on this panel (Figure 2-8). Insert panels come in two standard sizes—small (roughly 1 inch × 4 inches) and large (about 2 inches × 3 inches). The system I/O panel has apertures for four large inserts and two small inserts, or two large inserts and five small inserts (two of the large apertures can be converted to three small apertures).

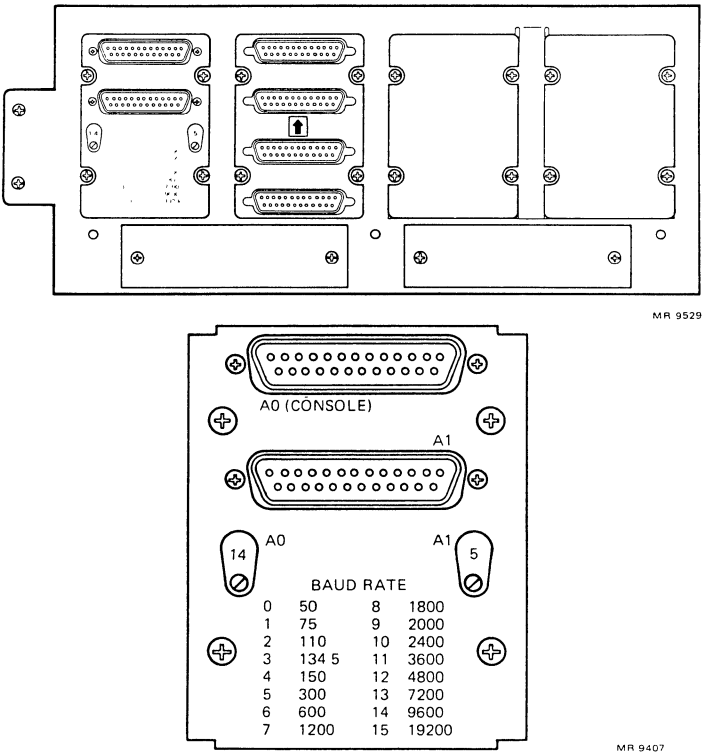


Figure 2-8 Detailed View of MICRO/PDP-11 Distribution Panel Insert

SYSTEM CONFIGURATIONS

The MICRO/PDP-11 chassis, as descibed above, is part of all MICRO/PDP-11 systems. Specific products described below vary in their mass storage components, amount of main memory, number of serial lines, and software license. With these basic system units and the options described later, you will be able to choose the configuration that best meets your needs. Configuration rules and worksheets are included in Appendix B.

All MICRO/PDP-11 systems come with a universal power supply that is switchable between 120 Vac and 240 Vac. The power cord and documentation are available in 15 different Country Kits. The kits are designed to meet location-specific language and power requirements and are sold at a nominal additional charge.

Floor-Mount (Tabletop) Systems

With diskette storage only:

11A23-F	Dual 400KB RX50 diskette drive, 256KB parity memory, two asynchronous serial lines. Can also be installed as a tabletop unit by removing the system base.

With 10MB Winchester and diskette storage:

11C23-F	10MB RD51 fixed disk, dual 400KB RX50 diskette drive, 256KB parity memory, two asynchronous serial lines. Can also be installed as a tabletop unit by removing the system base.
11C23-FA	Same as 11C23-F, with U.S.A. Country Kit.
SX-RA500-EX	Same as 11C23-F, but with six asynchronous serial lines and PDP-11 operating system license.
SX-RA500-FX	Same as 11C23-F, but with six asynchronous serial lines, PDP-11 operating system license and U.S.A. Country Kit.

Rack-Mount Systems

With diskette mass storage:

11A23-R	Dual 400KB RX50 diskette drive, 256KB parity memory, two asynchronous serial lines.

With 10MB Winchester and diskette mass storage:

11C23-R	10MB RD51 fixed disk, dual 400KB RX50 diskette drive, 256KB parity memory, two asynchronous serial lines.

Country Kits

Since language and power requirements vary, the MICRO/PDP-11 documentation and power cord are included in country kits sold separately.

BQ01-AA	U.S.A. and English-speaking Canada
BQ01-AC	French-speaking Canada

BQ01-AD	Denmark
BQ01-AE	United Kingdom
BQ01-AF	Finland
BQ01-AG	Germany
BQ01-AH	Holland
BQ01-AI	Italy
BQ01-AK	German-speaking Switzerland
BQ01-AL	French-speaking Switzerland
BQ01-AM	Sweden
BQ01-AN	Norway
BQ01-AP	France
BQ01-AS	Spain
BQ01-AZ	Australia

CPU PERFORMANCE OPTIONS

Three MICRO/PDP-11 options are available to enhance the processor's performance.

Floating-Point Performance Options

The standard KDF11-B CPU used in the MICRO/PDP-11 requires software subroutines for floating-point arithmetic.

Applications that make frequent use of floating-point data can benefit by adding PDP-11 floating-point instruction capability to the CPU hardware. MICRO/PDP-11 offers two ways to add floating-point instructions:

- KEF11-AA floating-point microcode expands the capability of the F-11 microprocessor by adding the microcode needed to implement PDP-11 floating-point instructions. KEF11-AA provides improved floating-point performance at low cost and mounts directly on the KDF11-B CPU module.
- FPF11 floating-point processor is an active logic unit optimized for executing the PDP-11 floating-point instructions. Working closely with the F-11 microprocessor, it monitors the instructions being fetched from memory. When it identifies a floating-point instruction, it assumes control over the F-11 microprocessor and executes the instruction, returning control to the F-11 on completion. FPF11 provides about six times the performance of the KEF11-AA at a moderate cost and requires a quad slot in the MICRO/PDP-11 backplane.

Character and Decimal-String Performance Option
The standard KDF11-B CPU used in the MICRO/PDP-11 requires software subroutines for character and decimal-string operations.

Applications using these data types can benefit by adding the character and decimal-string instructions to the CPU hardware with the KEF11-BB CIS microcode option, which extends the F-11 microprocessor's microcode. Since these data types closely align with those used in COBOL, these instructions are also collectively referred to as the commercial instruction set (CIS). KEF11-BB provides improved performance with these data types at a low cost and mounts directly on the CPU module.

PACKAGING OPTIONS
These options support the conversion of one style of MICRO/PDP-11 system packaging to another, in case your requirements change.

- BA23A-AF adapts MICRO/PDP-11 system chassis to tabletop or floor-mount installation.
- BA23A-AR adapts MICRO/PDP-11 system chassis for rack-mount installation.

MEMORY OPTIONS
The MICRO/PDP-11 system comes with 256KB or 512KB of parity memory. You can expand this to as much as 2MB by adding the following parity memory modules:

MSV11-PL	512KB, quad module, block-mode DMA
MSV11-PK	256KB, quad module, block-mode DMA
MSV11-LK	256KB, double module
MSV11-LF	128KB, double module

MSV11-PK and MSV11-PL are recommended for use in the MICRO/PDP-11 with the RD51 10MB Winchester disk because they support block-mode direct memory access (DMA) data transfers for reduced system loading and increased throughput. See the section "Special Features" in Chapter 7 for further information on block-mode DMA.

In some applications, it may be necessary to maintain data in memory when the system is powered down or during a power outage. Two CMOS memory modules, with an on-board battery, can maintain data when power is removed. Both are double modules.

MCV11-DC	32KB CMOS memory, 1180 hours typical, 100 hours worst case

MCV11-DA — 8KB CMOS memory, 2647 hours typical, 333 hours worst case

STORAGE OPTIONS

Standard MICRO/PDP-11 systems are available with the RX50 dual 400KB diskette drive or the RD51 10MB Winchester disk.

The following MICRO/PDP-11 storage options allow you to add storage capacity to a standard system, add a new device for media compatibility with other systems or with software distribution media, or to configure your own system.

Diskettes

Diskettes are small, removable, and inexpensive. The attributes make them suitable for personal removable media, backup, and data exchange. MICRO/PDP-11's primary diskette is the 5.25-inch RX50; the 8-inch RX02 is also available for media compatibility with other systems.

RX50-AA — Dual 400KB 5.25-inch diskette drive. Mounts in MICRO/PDP-11 system chassis. Requires RQDX1 controller (Figure 2-3).

RQDX1 — Controller module for RX50, RD51. Handles up to four logical units (RX50 = 2, RD51 = 1). Quad module, mounts in MICRO/PDP-11 backplane. Mass Storage Control Protocol (MSCP).

RXV21-EP — Dual 512KB 8-inch tabletop diskette drive and controller. Double controller module mounts in MICRO/PDP-11 system chassis (Figure 2-9).

Rigid Disks

Rigid disks provide more storage capacity and a lower cost per byte stored than diskettes. They are the real workhorses of storage system options. MICRO/PDP-11's main rigid disk is the 10MB fixed RD51 disk, which requires a removable mass storage device for backup and software transfer. The 10MB removable RL02 drive is also available for compatibility with other systems.

RD51-A — 10MB fixed Winchester disk. Mounts in MICRO/PDP-11 system chassis. Requires RQDX1 controller module (Figure 2-2).

RQDX1 — Described under "Diskettes" above.

RLV22-AP — 10MB RL02 removable disk drive and controller. Drive is 10.5-inch high, mounts in rack. Quad

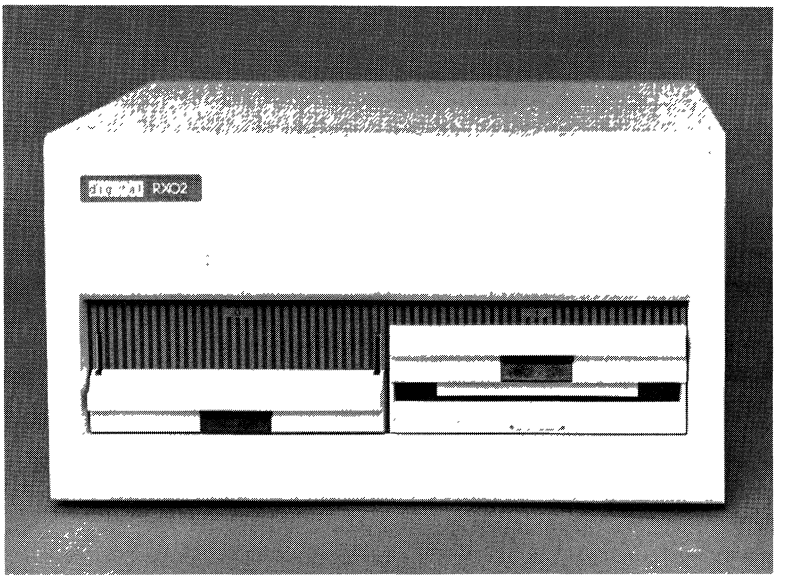


Figure 2-9 RX02

controller module mounts in MICRO/PDP-11 backplane, handles up to four drives (Figure 2-10).

RL02-AK — Add-on 10MB removable disk drive, rack-mount. Requires controller module (included in RLV22-AP).

Tapes

Magnetic tape storage gives up the disk's ability to directly access data in return for a very low cost per byte stored.

MICRO/PDP-11 tape storage offers the low-cost TU58 and the streaming, industry-compatible TSV05.

TU58-EB — Dual 256KB cartridge tape drive, tabletop subsystem. Interfaces to MICRO/PDP-11 through one of the asynchronous serial lines (Figure 2-11).

TSV05-BA — 9-track, 1600 bpi tape drive and controller. Twenty-five ips in TS11 emulation mode, 100 ips with user-supplied software. Drive is mounted in 40-inch cabinet, quad controller mounts in MICRO/PDP-11 backplane.

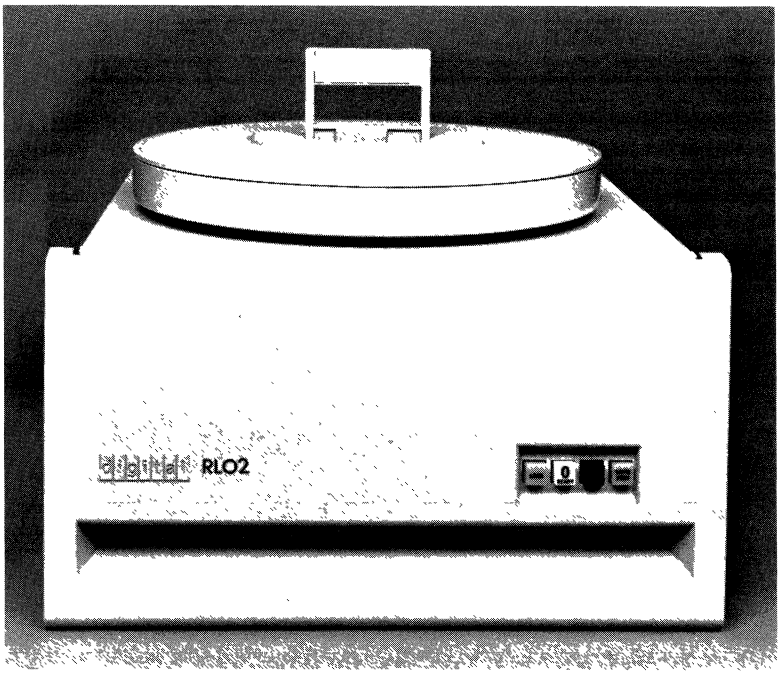


Figure 2-10 RL02/RLV22

COMMUNICATIONS INTERFACES

These interfaces are designed to allow MICRO/PDP-11 systems to communicate with user terminals and with other systems. In some cases, other devices will use the communications interfaces to interconnect with a system because the standards are so widely accepted (for instance, the TU58 cartridge tape drive uses an asynchronous serial line).

While this section simply describes the interfaces, Chapter 5 discusses them further in the context of networks and communications.

Asynchronous Interfaces

When you connect a terminal to a MICRO/PDP-11, whether locally or through modems and a telephone line, the connection will probably use asynchronous communication. Asynchronous serial lines are also used between the MICRO/PDP-11 and other devices where high speed and efficiency are not of primary importance.

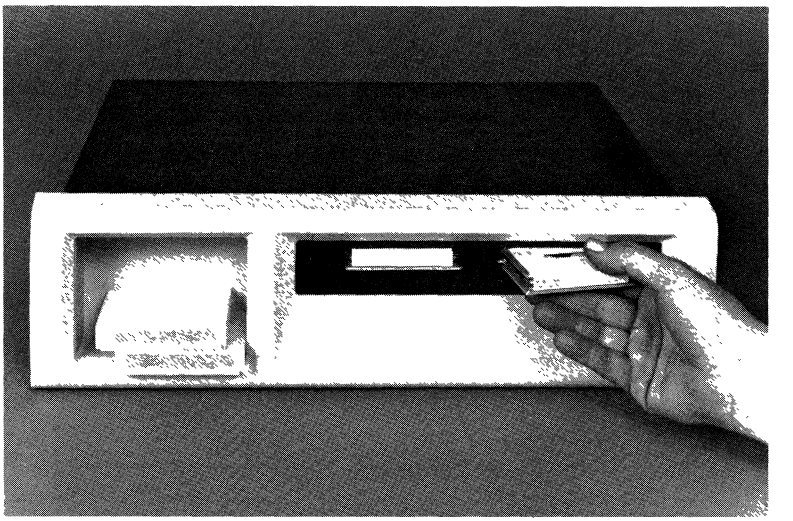


Figure 2-11 TU58

MICRO/PDP-11 offers a choice of three asynchronous serial-line interfaces. DLV11-EP is the lowest cost way to acquire a single serial line and has modem control. DLV11-JP offers the lowest cost per line and the highest number of lines per backplane slot. However, the DLV11-JP appears to the system as four separate devices with single-character buffers—incurring a software overhead penalty—and does not have modem control. The DZV11-CP is generally recommended for use in the MICRO/PDP-11. It appears to the system as a single device with a 64-character input buffer—resulting in more efficient I/O processing—and has modem control.

See Table 5-1 in Chapter 5 for a comparison of these devices' features.

DLV11-EP	Asynchronous serial-line interface. EIA RS-232C interface standard, modem control. Double module mounts in MICRO/PDP-11 backplane.
DLV11-JP	4 asynchronous serial-line interfaces. EIA RS-232C, RS-422, or RS423 interface standards, no modem control. Double module mounts in MICRO/PDP-11 backplane.

DZV11-CP — 4-line asynchronous serial multiplexer. EIA RS-232C interface standard, modem control. Quad module mounts in MICRO/PDP-11 backplane.

Synchronous Interfaces

Synchronous communication can be faster and more efficient than asynchronous communication. Synchronous communication is used to interact with another system or device intelligent enough to generate and interpret the protocol used to transmit the data.

MICRO/PDP-11 has three basic synchronous interface choices. DMV11 provides high-performance communication with other PDP-11, VAX-11, or DECsystem-10 or DECSYSTEM-20 systems equipped for DECnet's DDCMP protocol, with minimum loading on the MICRO/PDP-11 CPU. DPV11-DP is capable of handling bit-oriented or character-oriented synchronous protocols under software control. DUV11-DP is designed for character-oriented protocols only, under program control, and is offered primarily for continuity with preexisting software.

DMV11-AP — DDCMP synchronous serial-line interface. RS-232C, CCITT V.24, V.28 standards. Quad module mounts in MICRO/PDP-11 backplane.

DMV11-BP — Like DMV11-AP, but with CCITT V.35 interface standard.

DMV11-CP — Like DMV11-AP, but with integral modem for local connection with DMP11-AC/DMR11-AC (PDP-11 and VAX-11 systems) or with other DMV11-CP.

DMV11-FP — Like DMV11-AP, but with EIA RS423-A interface standard.

DPV11-DP — Bit-oriented (SDLC, HDLC) or character-oriented (BISYNC, DDLMP) synchronous serial-line interface. EIA RS-232C, RS-423 standard. Double module mounts in MICRO/PDP-11 backplane.

DUV11-DP — Character-oriented synchronous serial-line interface. EIA RS-232C standard. Quad module mounts in MICRO/PDP-11 backplane.

Ethernet Interface

Local area networks (LAN) offer high-performance communications among a variety of systems or other intelligent devices in a building or complex of buildings. MICRO/PDP-11 interfaces to the Ethernet local area network through the DEQNA Ethernet Controller and the H4000

Ethernet Cable Tap. The double module controller mounts in the MICRO/PDP-11 backplane, and the H4000 is installed remotely at the cable.

OTHER INTERFACES

Although the communications interfaces provide a widely accepted, understood means of connecting to a computer system, some applications require parallel or instrument bus interfaces.

Parallel Interfaces

These interfaces are called parallel because they have a large number of lines on which bits are transmitted simultaneously (in parallel) rather than having a single line on which bits are transmitted one after the other (in serial). They are also called "general purpose" interfaces because they offer the most direct way for an external device to access the data lines of the LSI-11 bus.

DRV11-LP	16 bits in, 16 bits out plus control signals. Program-controlled I/O. Double module, mounts in MICRO/PDP-11 backplane.
DRV11-BP	16 bits in, 16 bits out plus control signals. Direct memory access I/O. Quad module, mounts in MICRO/PDP-11 backplane.
DRV11-JP	64 bits plus control signals, organized as four 16-bit ports. Bit direction (input/output) program-selectable for each port. Program-controlled I/O. Double module mounts in MICRO/PDP-11 backplane.

Instrument Bus Interface

The IBV11-P interfaces the MICRO/PDP-11 to devices using the IEEE Standard 488 Instrument Bus. This double module mounts in the MICRO/PDP-11 backplane.

CHAPTER 3
TERMINALS

Your interaction with a computer takes place through a computer terminal; this makes the terminal more than just an add-on or afterthought. A well designed terminal can make the best of a mediocre system, but a poorly designed terminal can make an otherwise good system unpleasant to use.

DIGITAL's emphasis on bringing computers to people has naturally led to improvements in the computer terminal. Terminals designed and manufactured by DIGITAL are sold both as part of DIGITAL computer systems and by themselves for use with other manufacturers' equipment. They have been received so successfully that DIGITAL has become one of the largest terminal manufacturers in the world.

VIDEO TERMINALS

Video display terminals use a TV-like screen for output and a typewriter-like keyboard for input. *Alphanumeric* video terminals are capable of displaying characters—letters, numbers, and special characters—in a fixed format. *Graphic* video terminals can individually manipulate each picture element on the display screen and, therefore, can represent graphs, charts, and pictures. Typically, a graphic terminal can also emulate an alphanumeric terminal, but not vice versa. See Table 3-1 for a comparison of DIGITAL video terminals.

Alphanumeric Video Terminals

VT102—DIGITAL's VT100 is one of the most popular and imitated video terminals ever built. VT102 looks and functions like a VT100 with the Advanced Video Option and printer port, but the VT102 costs less.

The keyboard is separate from the display for operator comfort and is sculptured like a typewriter keyboard with a separate calculator-like 18-key numeric pad. The monitor displays 24 lines of 80 or 132 characters with individually selectable attributes—bold, blink, underline, or reverse. Double-width, double-height characters can be used for

Table 3-1 Comparison of Video Terminals

	VT102	VT101	VT125	RT102
Alphanumeric characters/line	80/132	80/132	80/132	80/132
Lines	24	24 (80 char/line) 14 (132 char/line)	24 (80 char/line) 14 (132 char/line) 24 (132 char/line)*	24
Graphics	—	—	Standard	—
Graphic resolution (pixels)	—	—	768 × 240	—
Character attributes	0–4	0–1	0–1 0–4*	0–4
Reverse video	Standard	Standard	Standard	Standard
Underline	Standard	Standard	Standard	Standard
Blink	Standard	—	Optional*	Standard
Build	Standard	—	Optional*	Standard
Smooth scroll	Standard	Standard	Standard	Standard
XON/XOFF	Standard	Standard	Standard	Standard
Communications	Full duplex Asymmetric Full duplex Half duplex	Full duplex	Full duplex	Full duplex Asymmetric Full duplex Half duplex
Modem control	Standard	—	—	Standard
Printer adapter	Standard	—	Optional	Standard

* With advanced video option VT1XX-AB.

legibility at a distance. Text can scroll smoothly up the screen rather than jump a full line at a time; the screen can be split into scrolling and nonscrolling regions. Many of the terminal's operating characteristics can be changed from the keyboard.

Asynchronous communication with the system can be full-duplex, full-duplex with asymmetric speeds, or half-duplex with modem control. Automatic synchronization codes (XOFF tells the computer to stop sending data, XON to start again) allow fast data rates without the possibility of lost information. VT102 can connect to a dedicated printer, such as an LA100 or LA50, to produce a hard-copy printout of the video screen.

VT102 functionality is also available in a rugged version, the RT102. For use in harsh environments, the RT102 has a sheet-steel case, heavy-duty filtration system, and a sealed keyboard.

VT101—VT101 is DIGITAL's low-cost video terminal in the tradition of the VT100. It provides 24 lines of 80 characters or 14 lines of 132 characters. Characters can be underlined or represented in reverse video. The display supports smooth scrolling and split-screen operation, and the operator can set terminal characteristics from the keyboard. Communications with the system are full-duplex asynchronous.

Graphic Video Terminals

VT125—The basic VT125 combines bit-mapped monochrome graphics and a printer port with VT100 functionality (Figure 3-1). The advanced video option (VT1XX-AB) increases the number of 132-character lines to 24 and allows selection of any combination of bold, blink, underline, and reverse video attributes on a character-by-character basis.

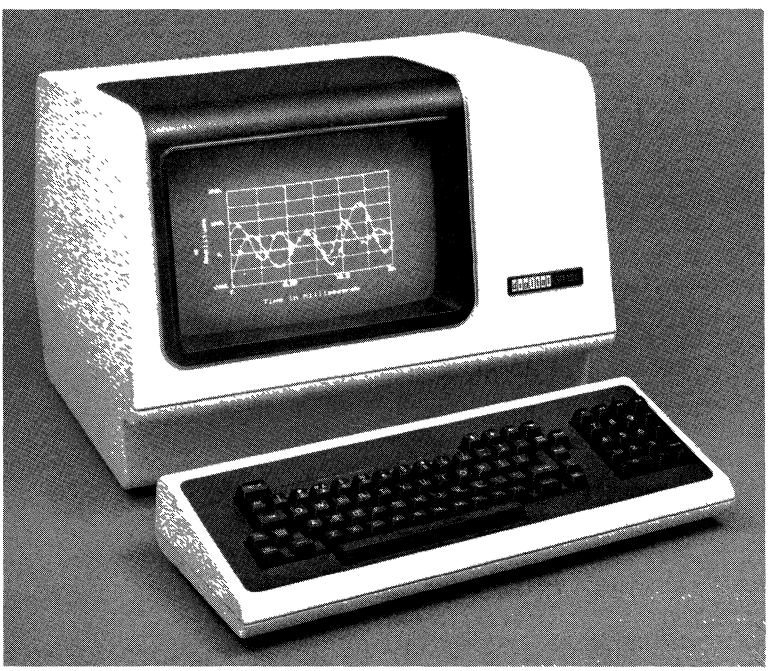


Figure 3-1 VT125

VT125 graphics are described by the Remote Graphics Instruction Set (ReGIS) which is executed by the VT125's microprocessor. ReGIS facilitates the high-level language interface to the VT125. ReGIS considers a picture to be a group of graphic objects where each object is a geometric shape that can be described by a few characters of information. For example, ReGIS understands that the shape of a circle applies to any circle that can be drawn and that it can be described on the screen by locating its center and a point on its circumference. The same kind of understanding for other graphic objects allows ReGIS to be a compact descriptor of graphic images.

The VT125's screen resolution is 768 (horizontal) × 240 (vertical). There are two graphics planes that can be displayed one at a time or at the same time. When displayed together, the two planes provide four monochrome intensities on the VT125 screen—black, white, and two shades of gray. When attached to an external color monitor, the VT125 can generate four colors from a palette of 64, using both graphics planes. The VT125 can connect directly to a graphics printer such as the LA100 or LA50 for a hard copy printout of the screen.

PRINTING TERMINALS

There are times when you really have to have it on paper; for these applications, your MICRO/PDP-11 can be used with DIGITAL's printing terminals. Refer to Table 3-2 for a comparison.

LA120 (DECwriter III)

The LA120 freestanding printing terminal is a sturdy, high-performance device with a reputation for reliability (Figure 3-2). It prints on multiple-copy, tractor-fed paper from 3 inches to 14.8 inches wide, at a maximum speed of 180 characters per second—fast enough to print a typical one-page memo in 20 seconds. Because of its 1K character buffer, bidirectional printing, and ability to skip quickly across spaces, the LA120 maximizes the throughput delivered to the paper.

The standard LA120 character set is U.S.A./United Kingdom. Character sets for Europe or the APL programming language are optional. Characters can be printed in eight sizes and in six choices of vertical line spacing. Characters are formed by an impact dot-matrix print head in a 7 × 7 dot format. Both keyboard send/receive (KSR) and receive-only (RO) versions are available.

LA100 (Letterprinter 100 and Letterwriter 100)

The LA100 printing terminals provide a lot of flexibility in a tabletop unit about the size of an electric typewriter (Figure 3-3). The Letter-

Table 3-2 Printing Terminals

	LA120 DECwriter III	LA100 Letterprinter 100 Letterwriter 100	LA12 Correspondent	LA50 Personal Printer	LQP02 Letter Quality Printer
Location	Freestanding	Desktop	Portable	Desktop	Desktop
Keyboard send/receive model	Yes	Yes	Yes	No	No
Receive-only model	Yes	Yes	No	Yes	Yes
Characters per line (max.)	216	216	132	132	158
Number of character sizes	8	8	8	5	Over 100 print wheels
Multiple fonts	—	Yes	—	—	Yes
Graphics	—	Standard	Standard	Standard	—
Paper feed	Tractor	Friction Tractor (Optional)	Friction Pin Tractor (Optional)	Friction Tractor	Friction Pin Tractor (Optional)
Print matrix	7 × 7	7 × 9 (draft mode) 33 × 9 (Optional memo mode) 33 × 18 (letter-quality mode)	9 × 9	7 × 9 (draft mode 13 × 9 (memo mode)	Full character print
Print speed (max. characters per second)	180	240 (draft mode) 30 (letter-quality mode) 80 (memo mode)	150	100 (draft mode) 50 (memo mode)	32
Integral modem	—	—	1200/300 baud (optional)	—	—

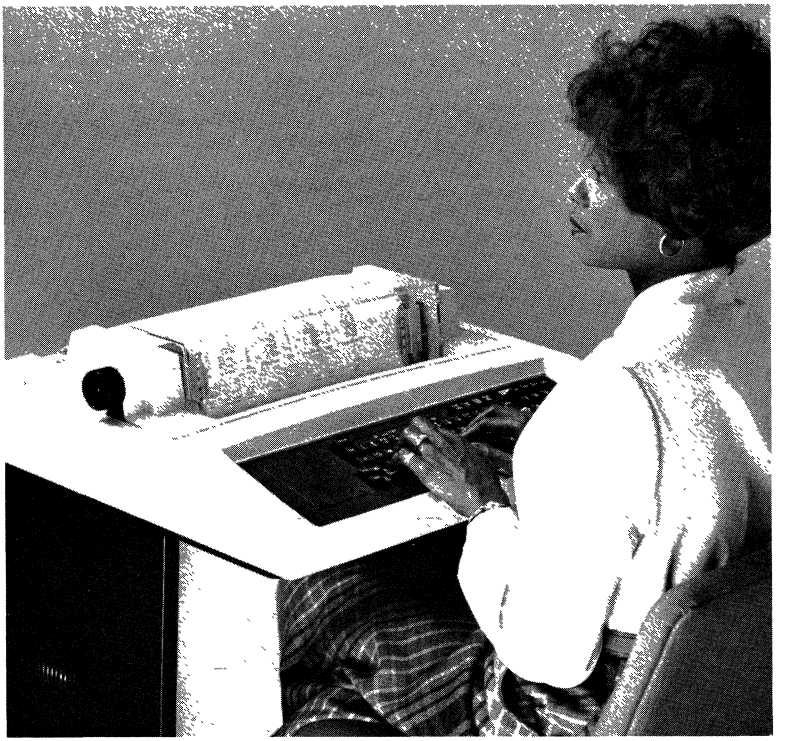


Figure 3-2 LA120

writer 100 is a keyboard send/receive (KSR) terminal, and the Letter-printer is receive-only. Both can print on friction-fed paper or, with an option, tractor-fed paper.

The impact dot-matrix printer offers you a choice of print quality and speed. In draft mode, it prints 240 characters a second maximum with a 7×9 print matrix. At the other extreme, letter-quality mode gives you a 33×18 print matrix at 30 characters per second. In between these modes is the optional memo-quality mode with its 33×9 print matrix and 80 character per second speed. Standard fonts included with the LA100 are Courier-10 and Orator-10. Optional fonts include Gothic-10, Symbol-10, Courier-12 and many others. Fonts can be selected by the system or from the keyboard.

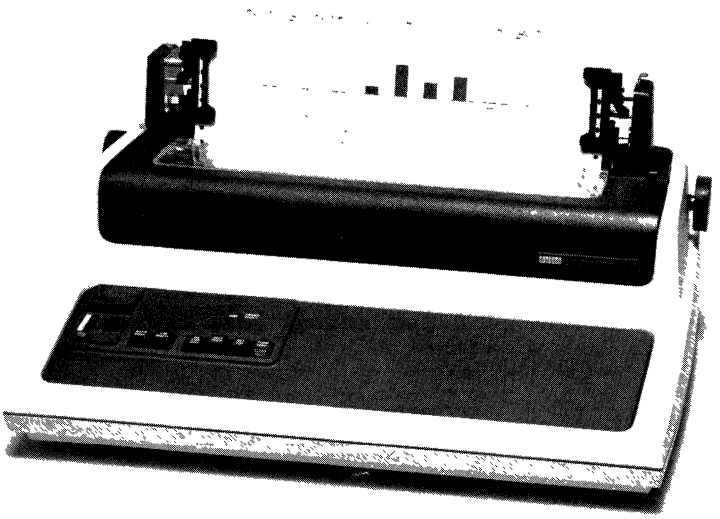


Figure 3-3 LA100 Send/Receive (KSR) and Receive-only

And the LA100 can print bit-map graphics as well. LA100 provides all of the formatting features found on the LA120—character size, vertical spacing, topform, tabs, and margins.

LQP02 Letter Quality Printer

When your output has to be top quality, the LQP02 receive-only printer gives you fully formed characters (Figure 3-4). The interchangeable daisy-wheel printing element allows you to select from over 100 different wheels and prints at 32 characters per second.

LQP02 prints on single sheets or fan-fold paper. An optional tractor allows bidirectional feeding of preprinted forms. And the optional dual-tray cut sheet feeder lets you alternate between letterhead and second sheets.

LA50 Personal Printer

Low-cost, tabletop printing is the LA50's strength (Figure 3-5). Its impact dot-matrix mechanism prints 7 × 9 matrix characters at 100

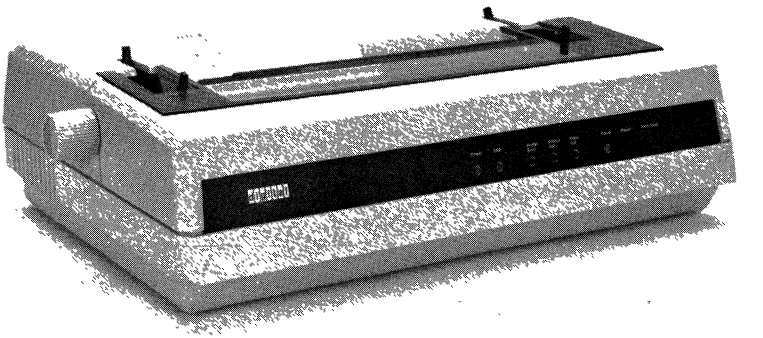


Figure 3-4 LQP02

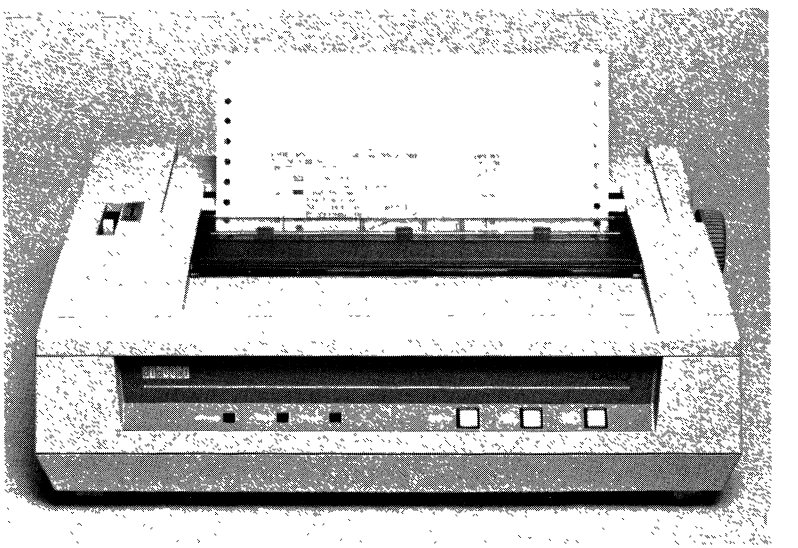


Figure 3-5 LA50

characters per second in draft mode, 13 × 9 characters at 50 characters per second in memo mode, as well as bit-map graphics.

Characters can be printed at pica (10) or elite (12) pitch, as well as compressed (16.5) for 132 characters on a line. Each pitch can also be printed in double-width characters. The LA50 has 250 printable characters, including:

- The standard 96-character ASCII set
- VT100 special character set
- Eight-bit multinational character set
- Eleven national character sets.

LA12 Correspondent

LA12 is DIGITAL's high-quality portable printing terminal, with integral modem and acoustic coupler (Figure 3-6). It prints up to 132 columns of 9 × 9 print dot-matrix characters at a maximum of 150 characters per second, as well as bit-map graphics. Unlike most portable teleprinters, the LA12 uses plain (not thermal) paper. Character sets and formatting features are similar to those of the LA120 and LA100.

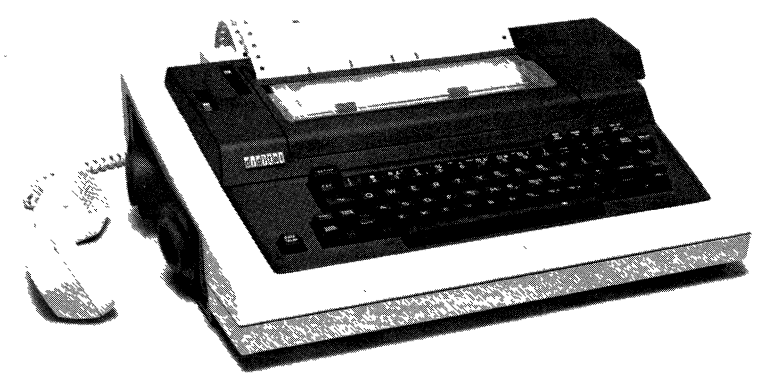


Figure 3-6 LA12

CHAPTER 4
SYSTEM SOFTWARE

The hardware components of a computer system—the processor, memory, storage, interfaces, and terminals—represent only the potential to do useful work until they are activated by software.

Realizing the hardware's potential in service to your application depends heavily on the efficiency of the software. Realizing potential also depends on how closely the software's functionality aligns with the application's requirements.

MICRO/PDP-11 inherits several advantages from its predecessors in the PDP-11 family:

- *Proven system software.* PDP-11 system software combines hundreds of programmer-years of development with thousands of programmer-years of user experience. PDP-11 software products have been refined and improved as a result of this experience.
- *Choice of system software.* There are six different families of PDP-11 operating systems, and you can choose the one that corresponds to your requirements.
- *Compatible system software.* PDP-11 system software is designed with the kind of compatibility that can help you. Higher level languages, file structures, and command language are, in many cases, compatible among PDP-11 operating systems and even with the VAX/VMS operating system.

OPERATING SYSTEMS

The operating system manages computer system resources. It allocates processor time, memory, and peripheral devices. It structures files so you can think of them logically rather than as device-dependent sets of data. It manages I/O to relieve you from the details.

There are six groups of PDP-11 operating systems. Each makes different tradeoffs in managing system resources. You can choose the one that is right for your application.

RSX-11 Family

RSX operating systems are designed to execute *multiple programs* concurrently. A program is allowed to execute (to use the CPU as a resource) until its immediate need for the CPU is completed or until an external event associated with a higher priority program takes place. If the higher priority program needs memory space, the lower priority program is swapped out to a disk. This ability to rapidly respond to external events makes the RSX operating system an appropriate choice where *real-time* reaction is important, for example, industrial process control or data acquisition. When tasks of equal priority are eligible to execute, a round-robin algorithm rotates their selection so that all receive an equal share of CPU time in the long run.

RSX operating systems are not limited to real-time tasks. They are designed to provide *login/logout* with passwords, access protection, user accounting, and other capabilities to support *multiple users* developing programs on the system. You can communicate with the operating system with easy-to-use DIGITAL Command Language (DCL) or with the traditional RSX Monitor Console Routine (MCR) command language.

As a program development user under RSX-11-PLUS or RSX-11M, you have the capability to use a broad range of DIGITAL high-level languages, software tools, and application programs. The RMS-11 file system will maintain your files in sequential, random, relative, or multikeyed ISAM (Indexed Sequential Access Method) organization.

Of the three members of the RSX family:

- RSX-11M-PLUS is generally recommended for MICRO/PDP-11 systems
- RSX-11M is available if you want to continue using it
- RSX-11S is a subset of RSX-11M designed to supervise the execution of memory-resident application programs.

RSTS/E and MICRO/RSTS

RSTS/E systems work *interactively* with people sitting at terminals. Each one of the *multiple users* can count on an almost immediate response to requests for access to programs, utilities and data, applications being run, and transactions in process.

As a user, you are associated with a job, and you interact with that job through your terminal. The *timesharing* job scheduler allows a job to

execute until its immediate need for the CPU is completed or until an allocated period of time expires. Then the eligible job with the highest priority is executed. If several eligible jobs have the same priority, a round-robin scheduler rotates their selection so each gets an equal share of CPU time.

You can also submit a *batch* job to run after working hours, for instance, and the batch processor job will supervise its execution in your absence.

Your files and file directory can be protected from unauthorized access by other users. You can even specify who can read a file and who is allowed to modify or update it.

You communicate with the operating system through the easy-to-use DIGITAL Command Language (DCL). Individual installations can add their own commands with the Concise Command Language (CCL). And with optional MENU-11/RSTS, you can build customized menus as a command interface for novice or infrequent users. BASIC-PLUS is included with the RSTS/E operating system.

MICRO/PDP-11 systems with the 10MB RD51 disk (and no additional rigid disks) should use Micro/RSTS rather than RSTS/E.

Micro/RSTS is an application-only subset of RSTS/E that leaves more of the RD51 disk for application programs and user data than RSTS/E would leave. An application developed on a host RSTS/E system—which could be a MICRO/PDP-11 with at least 20MB of storage or a larger PDP-11—can be executed on the target MICRO/PDP-11.

RT-11

The RT-11 operating system minimizes processor and storage demands, maximizing *efficiency* by targeting systems with a *single user.* This eliminates the overhead associated with job scheduling and access protection.

As the single user, you can develop programs or run batch programs as a background job while the system gives higher priority to a concurrent *real-time* foreground job. Your background program will run until an external event transfers control to the foreground job; the foreground job will run until its need for the processor is completed. Both the foreground and background jobs have access to all the system's resources, and they can communicate through files or a message transmission facility. For minimal overhead, there is also a single-job RT-11 monitor.

RT-11 is easy to use because of the DIGITAL Command Language (DCL) and the simplicity inherent in an operating system designed for

the single interactive user. On-line information about keyboard commands is available. And an automatic installation procedure installs the operating system simply by conducting an interactive dialog with you.

You can write programs without explicitly identifying the output device; you can defer selecting the device, for instance, until you run the program, directing printer output to the disk. When you add new devices to the system, your old programs can adapt easily.

Although RT-11 supports only one "command terminal," multiterminal support capability allows your programs to control up to 16 additional terminals.

CTS-300

CTS-300 is designed to support *commercial* applications. It consists of the RT-11 operating system, described above, plus DIBOL (DIGITAL's Business-Oriented Language) and a number of utilities. Most RT-11-dependent software products can also be run on CTS-300.

CTS-300, like RT-11, is a single-user system in the sense that there can be only one system command terminal. However, multiple terminals running multiple DIBOL jobs or developing multiple DIBOL programs are supported under the DIBOL run-time systems.

DIBOL has a Data Division and a Procedure Division, like COBOL, and provides the ability to manipulate data, evaluate arithmetic expressions, redefine records, call other programs, spool output, and access files.

Utilities included with CTS-300 include:

- DECform for video data entry. DECform defines screen formats, checks entered data for range and type, and totals and validates entered fields. It also supports additions, inquiries, changes, and verifications to DMS-300 files.
- DMS-300, a data management utility supporting sequential, random, and keyed ISAM files.
- Sort/merge.
- A line printer spooler utility that queues and manages files for printed output.

MicroPower/Pascal

MicroPower/Pascal describes two system environments: a host system that you use to create, build, and test *real-time application* software, and a target microcomputer system that runs the software.

The host system software includes an extended real-time Pascal compiler running under a subset of the RT-11 operating system. Your application programs can use MACRO-11 in addition to Pascal if necessary. A library of software modules for process synchronization, communication, scheduling, exception and interrupt handling, timer services, and device and file I/O in the target system makes it easier and faster to build your application.

Your application program is created and linked with the appropriate run-time software in the development system. It is then transported to the target system by down-line loading it over an asynchronous serial line, or by manually transferring it on removable storage media (i.e., diskette) or a read-only memory.

A symbolic debugging program runs in the development system. This program allows you to examine program variables and kernel data structures and to control the execution of the application running on the target machine.

DSM-11

DSM-11 is a complete *multi-user* system environment with *data management* capability and the interactive, high-level language MUMPS. With DSM-11, you can quickly write, test, debug, or modify a program to establish a working application.

The MUMPS language, originally developed at Massachusetts General Hospital, has syntax and semantics oriented toward solving data base related problems. A novice programmer can very quickly produce useful working code, although using the full range of MUMPS capabilities does require some programming experience.

MUMPS' text-handling capabilities allow the inspection of any data item for content (such as particular keywords) or for format (letters, numbers, or punctuation characters in a string of text)—capabilities useful for on-line data entry checking and correction. Text can be segmented or concatenated.

The DSM-11 hierarchical file structure allows you to design data file strategies to suit the needs of your particular environment. Dynamic file storage simplifies expansion or modification of the data base. The data base handler maintains an in-memory cache of disk data with write-through for high performance and data sharing.

UNIX

DIGITAL supports Bell Laboratories' UNIX Timesharing System Version 7 on the MICRO/PDP-11 under the name V7M-11. UNIX was designed to provide a simple, elegant, easy-to-use *interactive* programming environment to *multiple users.*

The hierarchical file system provides you with a directory of your files; you can create subdirectories to manage groups of files together. Input/output is performed by writing or reading a special file associated with an I/O device; this makes file and device I/O similar for ease of programming, allows a program to accept either a file or device without changes, and extends the file protection mechanism to I/O. As creator of a file, you can permit or deny read, write, and access protection to yourself and to other users.

Your interface to the operating system is through the Shell command line interpreter. Issuing Shell commands creates processes that can communicate via interprocess pipes, create subsidiary processes, and synchronize with these offspring processes.

V7M-11 languages include C, a high-level language conducive to structured programming; FORTRAN-77; and RATFOR, which adds a C-type control structure to FORTRAN; a BASIC-like interpretive language; and a programmable desk calculator. Software tools include a compiler-writing system, document-preparation programs, information-handling routines, and graphics support.

You can install V7M-11 yourself and run the UNIX System Exerciser Package to verify that it is functioning properly.

Summary

The key attributes of these operating systems are summarized in Table 4-1. This discussion is intended only as a general guide. An index to sources with more detailed information can be found in Appendix K. Also, refer to Appendix H for a summary of available software distribution media.

HIGH-LEVEL LANGUAGES

If you write programs for your MICRO/PDP-11, you will probably work more closely with a high-level language processor than with the operating system. In addition to simplifying the task of programming a given function, high-level language programs provide a measure of compatibility and transportability among systems. PDP-11 BASIC-PLUS-2, for instance, is a close subset of VAX-11 BASIC, and PDP-11 COBOL-81 shares a great deal of common syntax with VAX-11 COBOL.

BASIC

There are three variations of BASIC available for MICRO/PDP-11 systems:

- BASIC-11, optional for RT-11 and CTS-300 systems
- BASIC-PLUS, standard on RSTS/E systems

Table 4-1 Operating Systems: Key Attributes

	RSX-11M-PLUS RSX-11M RSX-11S	RSTS/E Micro/RSTS	RT-11 CTS-300	MicroPower/ Pascal	DSM-11	V7M-11 (UNIX)
Users	Multiple	Multiple	Single	Single	Multiple	Multiple
Terminals	Multiple	Multiple	Multiple	Multiple	Multiple	Multiple
Command Language	DCL,MCR	DCL	DCL	DCL	MUMPS	Shell
Job Scheduling	Event-driven Priority structured	Time-sliced Priority structured	Event-driven Foreground/ background	Event-driven	Timesharing	Timesharing
File System	RMS-11 Sequential Random Relative Multikey ISAM	RMS-11 Sequential Random Relative Multikey ISAM	Contiguous	Contiguous	Hierarchical	Hierarchical

- BASIC-PLUS-2, optional on RSX-11M, RSX-11M-PLUS, and RSTS/E systems.

All three versions use simple English commands, understandable abbreviations, and familiar symbols for mathematical and logical operations; all three are readily accessible to programmers who are not computer specialists. They have been used extensively in education, small businesses, laboratories, and for personal use.

BASIC-11 and BASIC-PLUS are interactive language processors. They provide immediate feedback which encourages practice and experimentation.

BASIC-PLUS-2 is a compiler that uses a superset of the BASIC-PLUS language with the ability to access RMS-11 files and to call external subroutines. On-line-help text makes BASIC-PLUS-2 easy to use. Since BASIC-PLUS-2 is a close subset of VAX-11 BASIC, the two languages can work together in a mixed VAX-11 and MICRO/PDP-11 environment.

C

C is a concise, expressive, structured program language designed to facilitate development. It is included with the UNIX operating system and is also available for the MICRO/PDP-11 system with RSX-11M-PLUS or RSX-11M.

COBOL-81

MICRO/PDP-11 systems with RSTS/E, RSX-11M-PLUS, or RSX-11M can program in COBOL-81 using an interactive high-performance compiler. COBOL is a language used extensively for business applications because of its orientation toward character, string, and file processing and for its compatibility with data processing procedures and audit trails. COBOL-81 provides ANSI 74 standard COBOL features. Performance can be improved by using COBOL-81 with the optional KEF11-BB Commercial Instruction Set option. COBOL-81 syntax is similar to VAX-11 COBOL; consequently, COBOL-81 programs can, in most cases, be compiled and executed on a VAX-11 without source code changes.

CORAL-66

CORAL-66 is the block-structured language prescribed by the British government for real-time and process control applications. Programs written in CORAL-66 are flexible and easy to maintain. CORAL-66 is available for MICRO/PDP-11 systems with RSX-11M PLUS and RSX-11M.

DIBOL

DIBOL, DIGITAL's Business-Oriented Language, is designed for creating interactive application programs. Like COBOL, it uses a data division and a procedure division with English-like procedural statements.

DECform, included with DIBOL, is an easy-to-use tool for designing video screen formats for data entry and file inquiry. Fields can be protected, automatically duplicated, totaled, checked, or validated.

DIBOL and DECform are standard with CTS-300 and optional with RSTS/E.

FORTRAN

Two FORTRAN compilers are available for MICRO/PDP-11 systems:

- FORTRAN IV for RT-11, RSX-11M, RSX-11M-PLUS, and RSTS/E systems
- FORTRAN-77 for RSX-11M, RSX-11M-PLUS, and RSTS/E systems.

FORTRAN-77 combines the efficient numeric computation for which FORTRAN is known with access to sequential file and relative and indexed files. This makes FORTRAN-77 ideal for writing software that must manipulate and perform calculations on structures of numeric data, as in accounting or statistical packages. FORTRAN-77 is built on the ANSI subset FORTRAN-77 1978 standard.

FORTRAN IV is a fast, one-pass optimizing compiler that implements an extended superset of the 1966 ANSI standard for FORTRAN. FORTRAN IV works efficiently in small memory environments and is capable of producing absolute binary code for standalone PDP-11 systems or for loading into ROM or PROM memory.

MUMPS

MUMPS is a language oriented toward data base applications. It is an integral part of DSM-11 and is described in the DSM-11 section above.

Pascal

Pascal, a block-structured language, is an integral part of MicroPower/Pascal and is described in the MicroPower/Pascal section above.

Summary

Table 4-2 summarizes the languages supported by each operating system. This table also identifies the languages that are able to take advantage of the processor performance upgrade options: KEF11-A floating-point microcode, FPF11 floating-point processor, and KEF11-

Table 4-2 Summary of Languages Supported by Operating Systems

	RSX-11M-PLUS RSX-11M RSX-11S[1]	RSTS/E Micro/ RSTS[1]	RT-11 CTS-300	Micro- Power/ Pascal	DSM-11	V7M-11 (UNIX)
BASIC[6]	—	—	X	—	—	X[2,3]
BASIC-PLUS[6]	—	X[3]	—	—	—	—
BASIC-PLUS-2[6]	X	X	—	—	—	—
C	X[4]	X[4]	—	—	—	X[3]
COBOL-81[7]	X	X	—	—	—	—
DIBOL	—	X	X[5]	—	—	—
FORTRAN IV	X	X	X	—	—	—
FORTRAN-77[8]	X	X	—	—	—	X[3]
MUMPS	—	—	—	—	X[3]	—
Pascal	X	—	—	X[3]	—	—

Notes:

1. Run-time only on RSX-11S, Micro/RSTS
2. BASIC-like interpretive language .
3. Included with operating system.
4. Available from DECUS (Digital Equipment Corporation Users' Society).
5. Included with CTS-300.
6. Can use KEF11-AA or FPF11 floating-point options .
7. Can use KEF11-BB CIS option.
8. Requires KEF11-AA or FPF11 floating-point option.

BB CIS (character and decimal string) microcode. An index to sources with more detailed information can be found in Appendix K. Also refer to Appendix H for a summary of available software distribution media.

SOFTWARE TOOLS AND APPLICATION-ORIENTED PROGRAMS

In addition to the high-level languages described above, there is a set of software tools and application-oriented programs that adds functionality of various kinds to the system. The operating systems compatible with each product are shown below in parentheses.

Communications software products are described in Chapter 5.

Refer to the *PDP-11 Software Source Book* for a complete list, including those software products developed by independent companies.

ADE (RSTS/E)—Development environment to enable you to use your departmental computer for your individual information processing function, without the help of a programmer.

BASIC Transportability Package (RSX-11M, RSX-11M-PLUS, RSTS/E)—Software and documentation aids for converting BASIC-PLUS and BASIC-11 programs to BASIC-PLUS-2.

DATATRIEVE (RSX-11M, RSX-11M-PLUS, RSTS/E)—Interactive query, report, and data maintenance system for the less sophisticated computer user.

DECgraph (RSTS/E)—High-level graphic capability for support of VT125 or other ReGIS devices.

DECmail (RSTS/E)—Electronic mail software.

DECtype-300 (CTS-300)—Word processing software package, with menu-driven document processing and list processing.

DECword/DP (RSTS/E)—Word processing software package, with menu-driven document processing and list processing.

DIBS (CTS-300, RSTS/E)—Multiuser, wholesale distributor's accounting package.

FMS-11 (RT-11, RSX-11M, RSX-11M-PLUS, RSTS/E)—Forms-oriented video I/O management system. Offloads interactive video I/O management tasks from application.

INDENT (RSTS/E)—Data entry and forms management product for commercial applications.

Instrument Bus Subroutines (RT-11)—Library of FORTRAN callable subroutines and device handler for the IBV11 IEEE bus interface.

MDE/T-11 (RT-11, RSX-11M, RSX-11M-PLUS)—In-circuit emulation development system for the MICRO/T-11 microprocessor.

MENU-11 (RSTS/E)—Adds simple, menu-driven user interface to RSTS/E applications.

QUILL (CTS-300)—Query and report generation package for use with existing files.

RGL-11 (RT-11, RSX-11M)—ReGIS graphics library routines for VT125 support. Callable from FORTRAN programs.

RTEM (RSX-11M, RSX-11M-PLUS)—Provides the RT-11 program development environment as one of the tasks in the RSX system.

SORT (RSX-11M, RSX-11M-PLUS, standard on RSTS/E)—Utility that sequences records of RMS-11 files.

SSP (RT-11, RSX-11M)—Library of over 100 mathematical and statistical subroutines. Written in FORTRAN.

CHAPTER 5
NETWORKS AND COMMUNICATIONS

INTRODUCTORY EXAMPLE

All computers process information, and they have to get it from somewhere. For a single system, the sources are obvious: the human user types at the terminal keyboard, a diskette containing the information is loaded into the system, or the system acquires data directly from a machine or instrument (Figure 5-1).

Imagine two standalone computer systems used by Department A and Department B of a small company. Both departments use the same external information (sales figures), and Department B also uses the output of Department A's computer (inventory levels) as input. Taking the brute-force approach, each department would enter the sales figures from the keyboard of their own computers; in addition, Department B would have to read the inventory levels from the screen of Department A's computer and type them in from the keyboard as well (Figure 5-2).

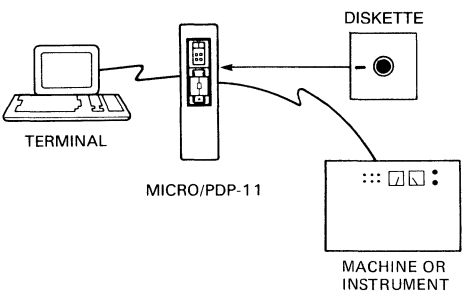


Figure 5-1 Standalone System

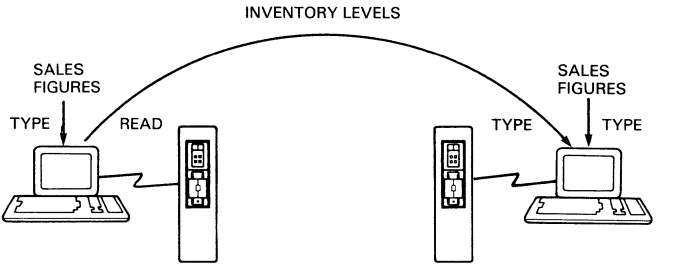


Figure 5-2 Two Standalone Systems. Department B reads information from Department A's screen and keys it into their own system.

To simplify things a bit, the two departments decide to cooperate. Department A enters the sales figures, processes the information, and then writes both the sales figures and the inventory levels on a removable diskette that Department B inserts in their computer instead of typing in all the numbers (Figure 5-3).

This is a definite improvement, but somebody from Department B still has to walk down the hall to Department A and pick up the floppy diskette. The fastest and easiest way to get the job done is to connect Department A's computer directly to Department B's computer. Then, as soon as the sales figures are entered by Department A, their computer can send them to B's computer (Figure 5-4). The inventory levels can be sent as soon as they're ready. There's no delay, and nobody from Department B has to walk down the hall.

Having put this simple *network* in operation at the main office, the people in both departments realize that the company's sales offices

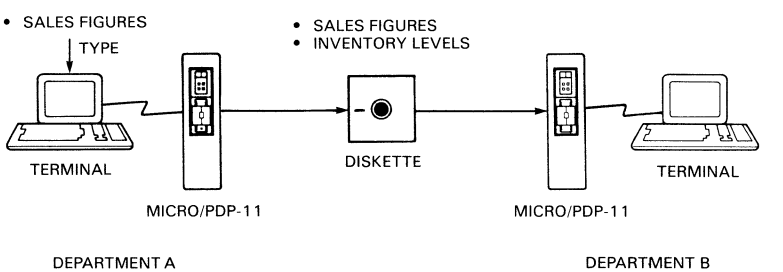


Figure 5-3 Two Standalone Systems. Information on Department A's system is given to Department B on diskette.

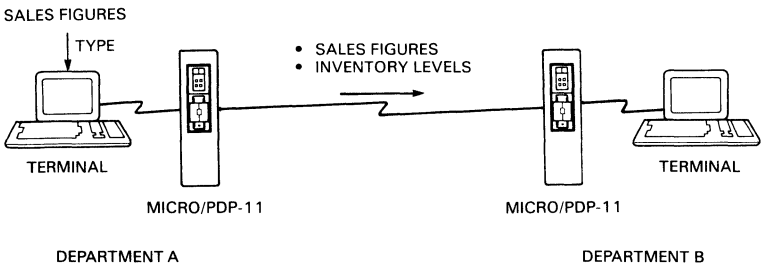


Figure 5-4 Two Networked Computer Systems

use personal computers to roll up their sales figures. Maybe they could be part of the network too.

The sales offices are in different cities and need to communicate for only a few minutes each day when they report their sales figures. Each sales office joins the network by telephone to the main office and transmits the sales figures with a *modem* connecting the computers at each end to the phone lines (Figure 5-5).

DIGITAL NETWORK ARCHITECTURE

The simple example above shows that there's nothing conceptually difficult about networks. Computers communicating directly with other computers can simplify, speed up, and organize the flow of data in an organization, with resulting improvements in productivity and control.

DIGITAL Network Architecture (DNA) is an integrated set of networking capabilities—hardware and software—that supports communication among DIGITAL systems and between DIGITAL systems and other manufacturers' equipment. DNA is structured like the International Standards Organization's layered model for Open Systems Intercommunication (Figure 5-6).

The lowest layer—Physical Link—governs the electrical and mechanical transport of information between connected systems. There are different ways to physically interconnect computers: cable, fiber optics, microwave satellite, or a switched network such as the telephone system. In addition to choosing a physical interconnection, it is also necessary to define how the signals on the physical connection represent binary data—whether they are sent synchronously or asynchronously, fast or slow. The Physical Link layer is embodied in hardware: cables, connectors, and the interface module.

Thanks to the Physical Link layer, the next higher layer, Data Link, does not have to contend with the electrical and mechanical details.

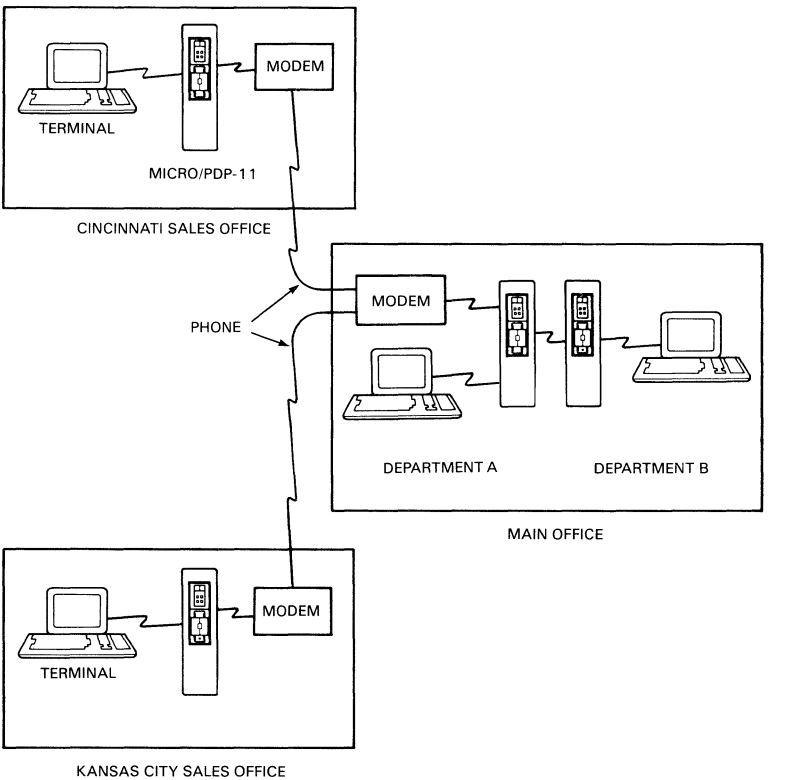


Figure 5-5 Four Networked Computer Systems

The Data Link layer can prepare messages for transmission according to the specified protocol, check the integrity of received messages, and manage access to the channel. Implementation of the Data Link layer is typically a joint hardware/software effort. In some cases, like a simple asynchronous interface, the hardware's contribution to the Data Link is modest. With other devices, such as the DEQNA Ethernet interface or the DMV11 DECnet interface, nearly all of the Data Link layer is implemented in hardware.

The Routing layer can rely on error-free connection to adjacent nodes thanks to the Data Link layer. This frees it to address messages, route them across intervening nodes, and control the flow of messages between nodes. The Routing layer, as well as higher layers, are implemented in software.

<table>
<tr><th>ISO SEVEN LAYERS</th><th>DNA LAYERS</th><th colspan="4">DNA FUNCTIONS</th></tr>
<tr><td rowspan="2">APPLICATION</td><td>USER</td><td colspan="4" rowspan="2">FILE TRANSFER
REMOTE RESOURCE ACCESS
DOWN LINE SYSTEM LOAD
REMOTE COMMAND FILE SUBMISSION
VIRTUAL TERMINALS</td></tr>
<tr><td>NETWORK MANAGEMENT</td></tr>
<tr><td>PRESENTATION</td><td>NETWORK APPLICATION</td></tr>
<tr><td>SESSION</td><td>SESSION CONTROL</td><td colspan="4" rowspan="2">TASK TO TASK</td></tr>
<tr><td>TRANSPORT</td><td>END COMMUNICATIONS</td></tr>
<tr><td>NETWORK</td><td>ROUTING</td><td colspan="4">ADAPTIVE ROUTING</td></tr>
<tr><td rowspan="2">DATA LINK</td><td rowspan="2">DATA LINK</td><td rowspan="2">DDCMP
POINT TO POINT
MULTIPOINT</td><td rowspan="2">X 25</td><td rowspan="2">ETHERNET</td></tr>
<tr><td>PHYSICAL</td></tr>
<tr><td></td><td>PHYSICAL LINK</td></tr>
</table>

Figure 5-6 DIGITAL Network Architecture's Layered Structure

Since the Routing layer establishes the path, the End Communications layer can address the end machine without concern for route and can perform end-to-end error recovery.

Session Control manages the system-dependent aspects of a communications session. For instance, when the End Communications layer reliably delivers a message from another manufacturer's system in the network, the Session Control layer interprets that message in a way the MICRO/PDP-11 system software is prepared to accept.

The Network Application layer converts data for display on terminal screens and printers.

The Network Management layer includes tools for monitoring network operation by logging events and collecting statistical and error information and for controlling network operation by tuning network parameters and testing nodes, lines, modems, and interfaces.

The User layer provides services directly supporting user and application tasks such as resource sharing, file transfers, and remote file access.

TYPICAL NETWORK CONFIGURATIONS

Systems can be interconnected in a network in a virtually unlimited number of ways; some of those ways are described here.

Each of the participating systems in a network is called a *node*. Two nodes communicate via a *link* or *connection*. For example, consider the network in Figure 5-7.

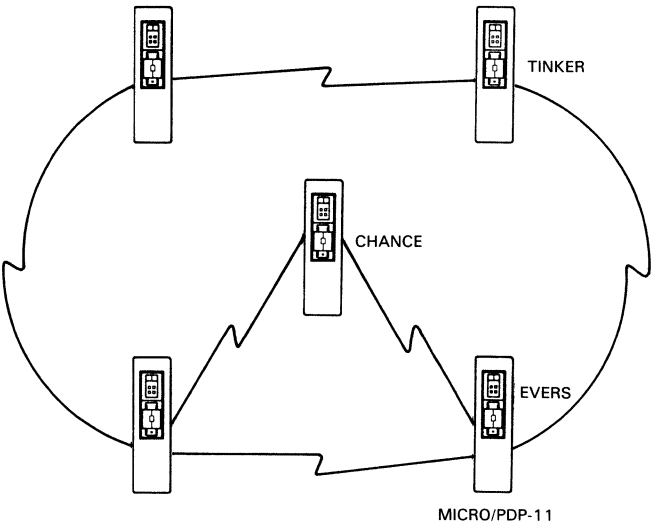


Figure 5-7 Example of Nodes Linked in a Network

The *physical links* are the circuits (cables, microwave, fiber optics, telephone lines, satellites, etc.) between two nodes (e.g., TINKER-to-EVERS). *Logical links* exist whenever two nodes can communicate, whether or not they are directly connected. There is no physical link TINKER-to-CHANCE in the network, but there is a logical link if EVERS is capable of *routing*, or passing along, a message from TINKER intended for CHANCE. EVERS will have to decide how to handle these routed messages, and it will take longer for a message to go from TINKER to EVERS to CHANCE than it would from TINKER directly to CHANCE. But there are fewer physical links, and physical links cost money.

As the number of nodes increases, the cost of physical links becomes more important because the number of physical links required for a *fully connected* network goes up very rapidly (Figure 5-8).

Another way to reduce the number of physical links is with a *multidrop* or *multipoint* link. In this case, more than two nodes connect to the same physical link (Figure 5-9). Each node has to determine which messages are intended for it and must manage access to the link to avoid conflicts.

In a network with *centralized control*, one node—a mainframe computer, for instance—decides which nodes can send messages, when,

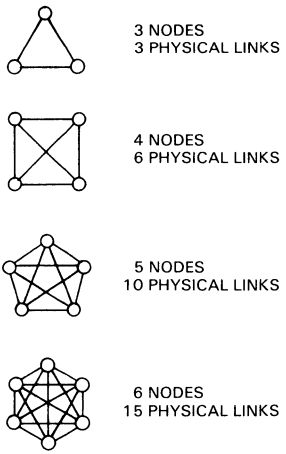


Figure 5-8 Examples of Fully Connected Networks. As the number of nodes increases, so does the number of physical links.

and for how long. In a network with *distributed control*, each node understands a set of rules that allow each to access the network independently.

A *star* network has a clearly designated central node, with all outlying nodes physically linked to it (Figure 5-10). This approach is efficient when most of the communication is between the central node and one of the outlying nodes, as in a timesharing network or shared-processor word processing system. But all messages between outlying nodes must go through the central node—a heavy burden if there is a lot of traffic. And the central node is a single point of failure; if it isn't working, neither is the network. Network control is often centralized in the central node, but distributed control is possible.

In a *ring* network, each node is physically linked to two adjacent nodes, and the physical links form an unbroken circle (Figure 5-11). Messages circulate around the ring, and each node has to retransmit messages not addressed to itself. This is easier than generalized

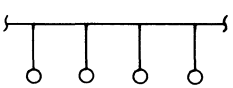


Figure 5-9 Multipoint Link

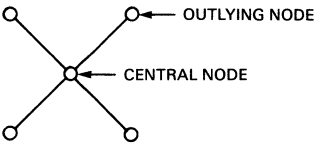


Figure 5-10 Star Network

routing, however, since the only place to retransmit the message is to the next node in the ring. One way of implementing distributed control in a ring network is through *token passing*. A special message called the token circulates around the ring; a node can claim access to the network by grabbing the token as it passes through.

The *bus* network configuration is similar to a multidrop link (Figure 5-12). Messages placed on the shared physical link reach all nodes, and the intended receiver must recognize the message's address in order to receive the transmission. None of the nodes, however, have to route or retransmit messages intended for other nodes. A bus network typically uses distributed control. There is no single point of failure. The *Ethernet* local area network is a bus configuration.

An *unconstrained* network doesn't look like any of the above network categories (Figure 5-13). Typically, the placement of physical links is

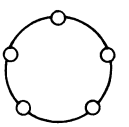


Figure 5-11 Ring Network

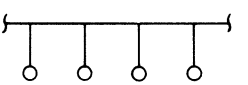


Figure 5-12 Bus Network

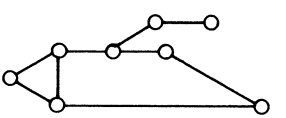


Figure 5-13 Unconstrained Network

based on tradeoffs between connection costs, traffic level, and the required reliability. Some, but not necessarily all, of the nodes have to have the capability to route messages addressed to other nodes. Long-distance *packet-switched* networks are often unconstrained.

TYPES OF LINKS

MICRO/PDP-11 offers a variety of interfaces supporting various implementations of the physical link and data link levels of DNA.

Ethernet

Computer systems have become more accessible. Increasing numbers of them—departmental systems, word processors, professional workstations, and personal computers—have found their way into the workplace. In addition, low-cost microprocessors are being designed into products that are not primarily considered as computer systems: copiers, telephones, private branch exchanges, facsimile machines.

It makes sense that an organization can operate more efficiently and productively if all this intelligent equipment can be interconnected. The concept of a local area network is based on the "80/20 rule": 80% of the information used in a local environment is generated within the environment and only 20% is brought in from the outside; 80% of the generated information is used within the local environment and only 20% is sent outside.

On this assumption, local area networks, like Ethernet, trade off long-distance capability in order to provide high speed and bandwidth, low cost, compatibility, flexibility, extendability, high reliability, and ease of maintenance. They allow a great number and variety of machines to exchange large amounts of information at high speed over limited distances.

The Physical Link and Data Link layers of Ethernet are implemented on the MICRO/PDP-11 by the DEQNA Ethernet Communications Controller, the H4000 Ethernet Transceiver and Cable Tap, and a transceiver cable connecting the two (Figure 5-14).

The Ethernet cable operates at 10 megabits per second, and each network segment can include up to 100 transceivers on a 500-meter cable (Figure 5-15). Segments can be interconnected via repeaters into a larger network with up to 1024 nodes (Figure 5-16). Segments separated by up to 1000 meters can be interconnected by a remote repeater (Figure 5-17).

The access control method is called Carrier Sense, Multiple Access with Collision Detection (CSMA/CD). Any node that wants to transmit listens for a pause between packets on the cable; the node then begins to transmit while continuing to listen. If another node has

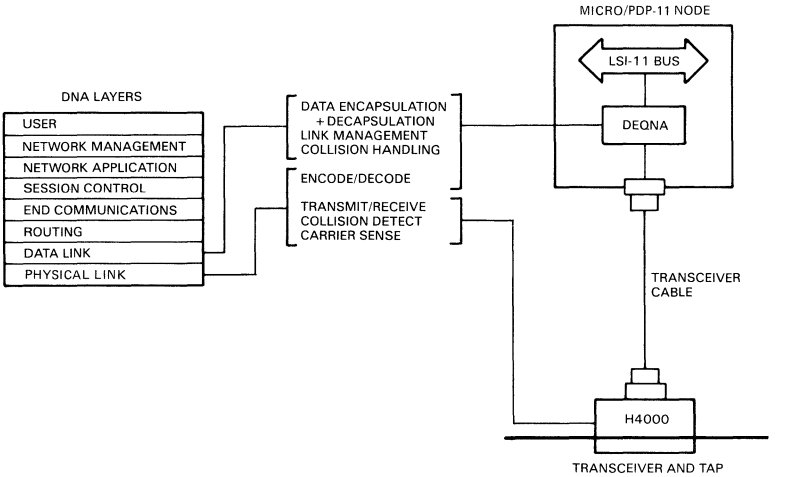


Figure 5-14 Implementation of Physical Link and Data Link Layers on Ethernet

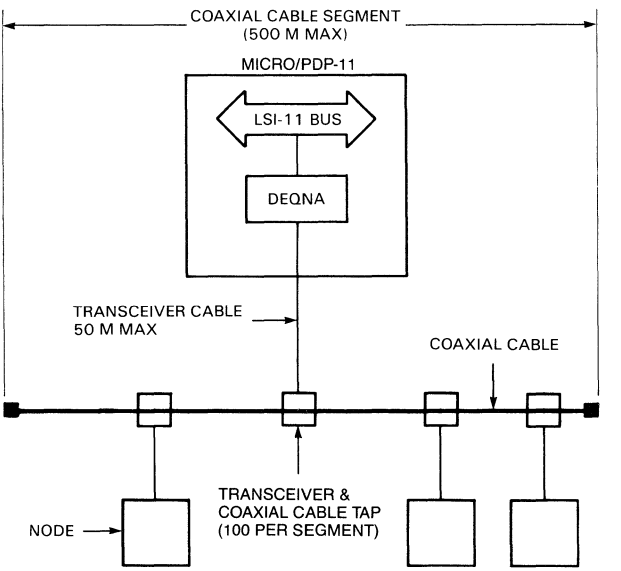


Figure 5-15 A Small Ethernet Configuration

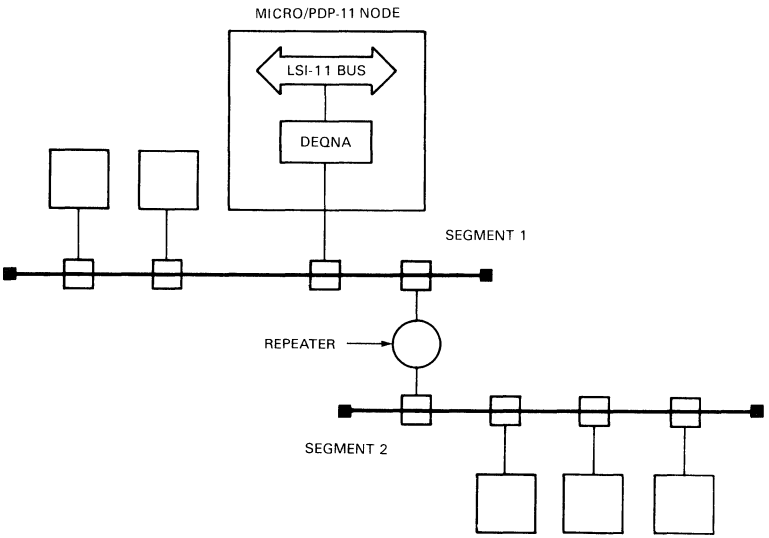


Figure 5-16 A Medium-Scale Ethernet Configuration

started to transmit at the same time, they both stop transmitting and wait for a random time interval before trying to transmit again.

The H4000 Transceiver and Tap connects to the Ethernet cable and can be installed on an operating cable with little or no disruption to traffic. The transmit/receive, carrier sense, and collision detection functions of the Physical Link layer are implemented in the H4000 Transceiver.

A transceiver cable, which can be up to 50 meters long, connects the H4000 Transceiver to the DEQNA Ethernet interface in a MICRO/PDP-11.

The DEQNA Ethernet interface handles the remaining function of the Physical Link layer by encoding and decoding the data exchanged with the H4000 Transceiver. In addition, it implements the functions of the Data Link layer by encapsulating and decapsulating messages, handling collisions, and managing links.

Where the DEQNA Ethernet interface leaves off, DECnet software picks up the Routing and higher layers of the DIGITAL Network Architecture. Ethernet links via the DEQNA are supported on the

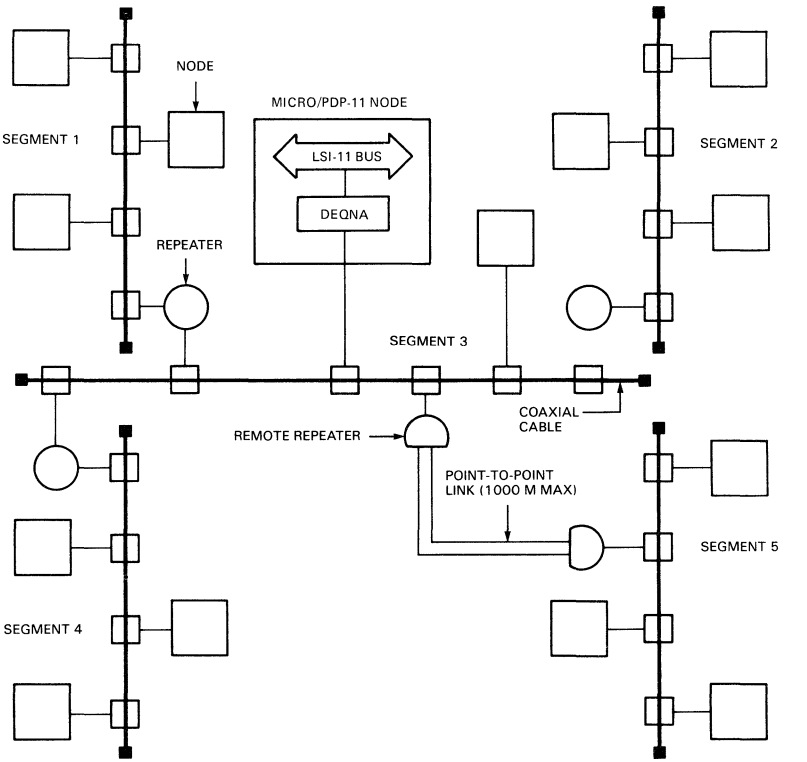


Figure 5-17 A Large-Scale Ethernet Configuration

MICRO/PDP-11 by DECnet-11M, DECnet-11M-PLUS, and DECnet-11S.

Up to eight Ethernet nodes can be connected to a single H4000 Transceiver and Tap, using the DELNI Ethernet Concentrator (Figure 5-18). The Ethernet nodes can include MICRO/PDP-11s with the DEQNA Ethernet interface or VAX-11s and UNIBUS PDP-11s with the DEUNA Ethernet interface. This interconnection takes place at the Physical Link level, and is transparent to the DECnet software.

For localized interconnections where an Ethernet segment is unavailable or unnecessary, the DELNI provides interconnection among up to eight Ethernet nodes without connection to an Ethernet cable (Figure 5-19).

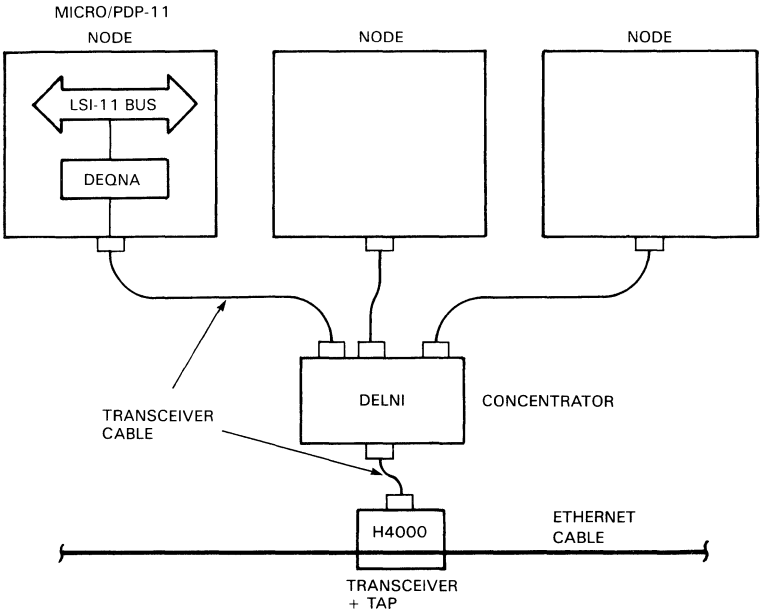


Figure 5-18 Ethernet Nodes Connected to H4000 Transceiver and Tap via DELNI Ethernet Concentrator

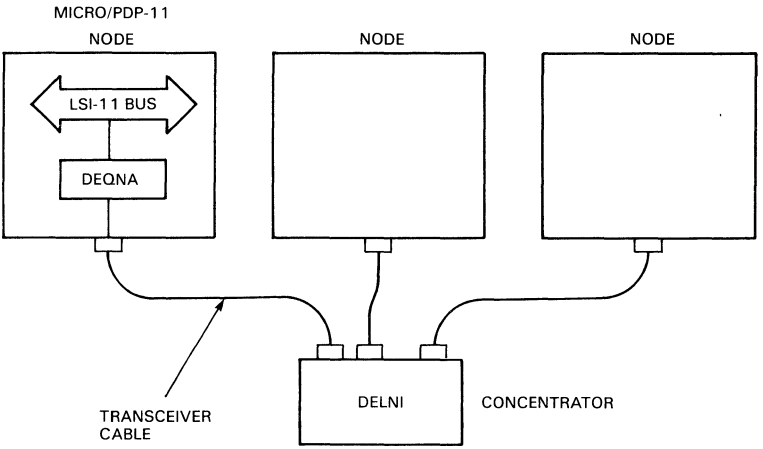


Figure 5-19 Localized Interconnection of Ethernet Nodes Using DELNI

Asynchronous Links

Asynchronous links provide a flexible, generally accepted, and time-honored means of interconnecting nodes separated by large or small distances where high speed and efficiency are not of primary importance. Data over an asynchronous link is character-oriented; characters can be 5 to 8 bits long, preceded by a start bit, and followed by stop bits. Any length of time can elapse between the stop bits of one character and the start bit of the next. The presence of the start and stop bits, which communicate no data, and the potential empty intervals between characters reduce the efficiency of asynchronous links in communicating data.

The physical link can take one of two main forms:

1. EIA Standard Signals, Long Distance (Modem Connection)—An asynchronous interface module can be connected to a modem which allows the data to be transmitted over telephone lines. The phone lines can be private, leased, or part of the public switched network. At the other end of the telephone line must be a compatible modem attached to an asynchronous device, such as a terminal or another interface module (Figure 5-20).

   The electrical and mechanical characteristics of the interface-to-modem connection are defined by various standards, including those of the Electrical Industries Association (EIA RS-232C, RS-422, RS-423) and the International Consultative Committee on Telegraphy and Telephony (CCITT).

2. EIA Standard Signals, Local Connection—Over short distances, an interface module generating EIA-standard signals can be connected to another EIA device—computer system, terminal, etc.—without the use of intervening modems. In place of the two modems and the telephone line is a *null modem cable* which connects each EIA device's transmit signal to the other's receive and which simulates the necessary control signals (Figure 5-21).

MICRO/PDP-11 systems come with two EIA connections as part of the CPU module. Some models include four additional connections as part of the DZV11-CP multiplexer. Optional interface modules for adding more asynchronous serial lines are:

- DLV11-EP, a single EIA standard line
- DLV11-JP, four EIA standard lines
- DZV11-CP, four EIA standard lines mutiplexed to a single interface controller and input buffer to reduce system loading and improve data integrity.

The characteristics of these devices are compared in Table 5-1.

Table 5-1 Comparison of Asynchronous Serial Interfaces

	DLV11-EP	DLV11-JP	DZV11-CP
Number of lines	1	4	4
Module size	Double	Double	Quad
Interface standards	EIA RS-232C CCITT V.24	EIA RS-232C EIA RS-422 EIA RS-423	EIA RS-232C
Modem control	Yes	No	Yes
Operating systems	RSX RSTS/E RT-11 DSM-11	RSX RSTS/E DT-11 DSM-11	RT-11 RSX RSTS/E UNIX
Network software	DECnet-RT DECnet-11M, -11S, -11M-PLUS DX/11M DX/RSTS	DX/11M DX/RSTS	DECnet-11M -11S, -11M-PLUS DX/11M DX/RSTS

Synchronous Links

For communicating over distances longer than a local area network, where speed and efficiency are important, *synchronous* links are appropriate. Synchronous communication sends a *block* of characters enclosed in a *frame*. The exact content of the frame varies from one *protocol* to another, but typically consists of text, identification of the beginning and the end of the frame, and information ensuring reliable reception of the text. Since the amount of extra information needed to complete the frame is fixed, the efficiency of synchronous transmission increases as the size of the text block increases.

Some synchronous protocols, such as DIGITAL's DDCMP and IBM's Bisync, require the length of the text to be an integral number of characters or bytes; these are called *character-oriented protocols*. Others, including IBM's SDLC and the International Standards Organization's HDLC, allow the text length to be any number of bits; these are referred to as *bit-oriented protocols*.

The physical link can take one of three basic forms:

1. Modem Connection, Long Distance—For synchronous communication over distance, the interface module is connected to a modem. The modem connection can be specified by one of the EIA standards (RS-232C, -422, or -423) or by a CCITT standard

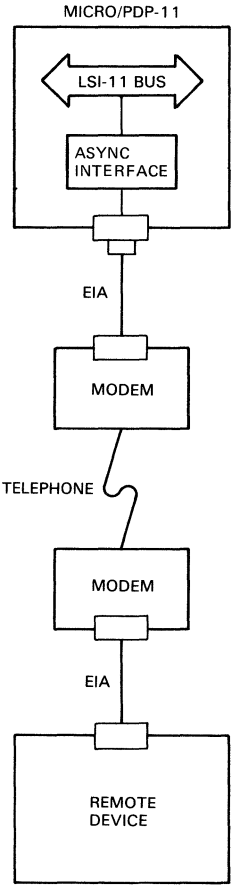


Figure 5-20 MICRO/PDP-11 Asynchronous, Physical Link: Long-Distance Connection

(V.24, V.28, or V.35). The modem itself can connect to a private line, a leased telephone line, or to the public switched telephone network. The choice of modem and the line connecting the modems will depend on the speed of the data communication (Figure 5-22).

2. Modem Eliminator, Short Distance—Connecting two local synchronous devices that interface via the EIA or CCITT standards requires a modem eliminator, which looks like a modem to each device. The definition of "short distance" depends on the speed

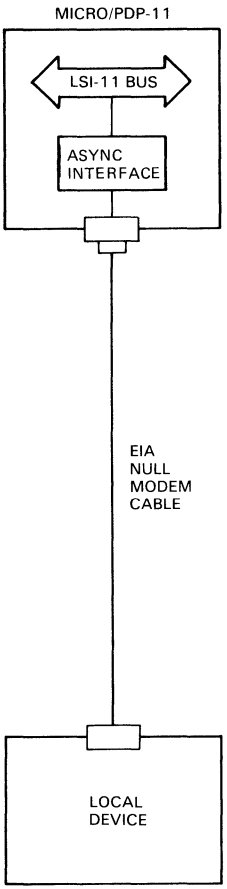


Figure 5-21 MICRO/PDP-11 Asynchronous, Physical Link: Local Connection

and other factors but could be from several hundred feet to a few miles (Figure 5-23).

3. Integral Modem, Local Connection—The MICRO/PDP-11's optional DMV11 interface can make use of a self-contained modem that is compatible with other DIGITAL DDCMP interfaces in both point-to-point and multidrop configurations (Figure 5-24).

The communications protocol, part of the Data Link layer, is implemented in the hardware of some interface modules; for other interface modules, this protocol is provided by communications software.

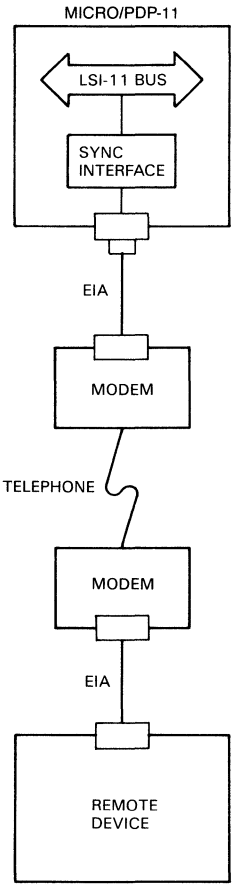


Figure 5-22 MICRO/PDP-11 Synchronous, Physical Link: Long-Distance Connection

Synchronous interface modules available from DIGITAL for the MICRO/PDP-11 are:

- DPV11-DP, a single synchronous line for bit-oriented or character-oriented protocols
- DUV11-DP, a single synchronous line for character-oriented protocols only
- DMV11, a single synchronous line for the DDCMP protocol only, in three variations of interface standards.

See Table 5-2 for a comparison of these interface modules.

Table 5-2 Comparison of Synchronous Serial Interfaces

	DPV11-DP	DUV11-DP	DMV11-AP	DMV11-BP	DMV11-CP	DMV11-FP
Number of lines	1	1	1	1	1	1
Multidrop links	No	No	Yes	Yes	Yes	Yes
Module size	Double	Quad	Quad	Quad	Quad	Quad
I/O method	Program control	Program control	DMA	DMA	DMA	DMA
Interface standards	EIA RS-232C EIA RS-422A EIA RS-423A	EIA RS-232C	EIA RS-232C	CCITT V.35	Integral modem	RS-423A
Software-driven protocols	Bit or character	Character	—	—	—	—
Integral protocol	—	—	DDCMP	DDCMP	DDCMP	DDCMP
Operating systems	—	RSX	—	—	—	—
Network software						
DECnet	DECnet-RT DECnet-11M, -11S, -11M-PLUS	DECnet-RT DECnet-11M, -11S	DECnet-RT DECnet-11M, -11S, -11M - PLUS DECnet/E	DECnet-RT DECnet-11M, -11S, -11M- PLUS DECnet/E	DECnet-RT DECnet-11M, -11S, -11M- PLUS DECnet/E	DECnet-RT DECnet-11M -11S, -11M- PLUS DECnet/E
Internet	RSX-11 2780/3780 protocol emulator	RSX-11 2780/3780 protocol emulator RT-11 2780/3780 protocol emulator RSX-11 3271 protocol emulator MUX200/RSX	—	—	—	—
Packetnet	RSX-11 PSI	—	—	—	—	—

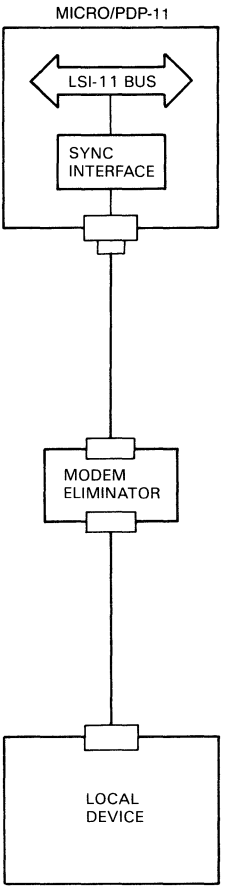


Figure 5-23 MICRO/PDP-11 Synchronous, Physical Link: Short-Distance Connection

NETWORK SOFTWARE

After the network links are established, communications software completes the network functionality within the DNA framework.

DIGITAL's network software is broadly divided into three categories: DECnet, for communication among DIGITAL systems; Internet, for communication with other manufacturers' equipment; and Packetnet, for communication with other participants in public packet-switched networks.

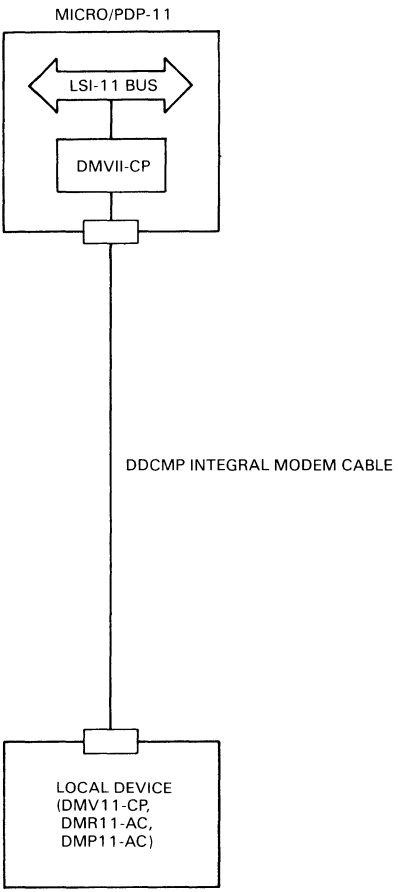


Figure 5-24 MICRO/PDP-11 Synchronous Link with Optional DMV11 Interface: Local Connection

DECnet

DECnet software supports communication among DIGITAL computer systems—MICRO/PDP-11, other 16-bit PDP-11s, 32-bit VAX systems, and the 36-bit DECsystem-10s and DECSYSTEM-20s. Data on the physical links is independent of system type, and the DECnet software converts the received data into formats that the operating system is prepared to accept. Because of this interdependence with

the operating system, there are five versions of DECnet available for the MICRO/PDP-11:

- DECnet-RT for use with the RT-11 operating system
- DECnet-11M for use with the RSX-11M operating system
- DECnet-11M-PLUS for use with the RSX-11M-PLUS operating system
- DECnet-11S for use with the RSX-11S operating system
- DECnet/E for use with the RSTS/E operating system.

Tables 5-1 and 5-2 list the versions of DECnet that support each of the MICRO/PDP-11's interface modules.

DECnet lets you put the network to full use by providing higher level network functions. Following are the key elements:

- *Task-to-task* communication allows programs executing in different systems, under different operating systems, and written in different languages to exchange information.
- *File transfer* supports the exchange of sequential ASCII or binary files between different operating systems.
- *Remote file access* lets you read, write, or modify files on another system.
- *Remote command file submission and execution* allows one computer system to direct another to execute commands which can be resident on the remote system or sent as part of the request.
- *Down-line loading* allows programs developed on a system with appropriate peripherals and resources to be transmitted to another system (such as a small, memory-only system) for execution.
- *Network command terminal* gives a terminal user logical connection to a remote system with the same operating system; the terminal operates as if it were directly connected to the remote system.
- *Network management* provides the tools for monitoring and controlling network operation.

Not all of these features are supported in each operating system's version of DECnet. See Table 5-3 for a cross-reference.

Personal Computer Interconnection

Each of DIGITAL's three personal computers—the Professional 300 series, DECmate II, and Rainbow 100—can be connected to the MICRO/PDP-11 over an asynchronous line. Although this communication does not use DECnet software, it does fall in the DIGITAL-to-DIGITAL communication category.

Table 5-3 DECnet Products

Capability	-RT	-11M	-11S	-11M-PLUS	/E
Program-to-program	X	X	X	X	X
Network command terminal[1]	X	X	X	X	X
File transfer	X	X		X	X
Command/batch file submission	X[2]	X[3]		X[3]	X
Command/batch file execution	X[2]	X	X		X
Remote file access	X	X	X[4]	X	X
Down-line system loading		X		X	
Down-line task loading		X		X	

Notes:

1. Terminals on these systems can log onto other systems running the same operating system. DECnet-11S does not support connection from remote command terminals.
2. Requester-only function.
3. Cannot submit command/batch files to DECnet-RT, DECnet/E or DECnet-VAX systems. However, DECnet/E can execute files already at the DECnet node.
4. Offers local users network access to remote file systems. Does not allow users on remote systems to access local files.

Two modes of communication are supported:

- File transfer supervises the transmission of an entire file between a personal computer and a MICRO/PDP-11 without operator intervention.

  For file transfer with a Professional 300, the MICRO/PDP-11 must have the Host File Transfer Package running under RSX-11M or RSX-11M-PLUS. The Professional 300 must have PRO/Communications software (Figure 5-25).

For file transfer with a DECmate II or a Rainbow 100, the MICRO/PDP-11 must have DX/11M running under RSX-11M or RSX-11M-PLUS, or DX/RSTS running under RSTS/E or Micro/RSTS. DECmate II must have the WPS-8 Communications Package. Rainbow 100 must have the VT102 or VT125 Communications Package (Figure 5-26).

- Terminal emulation provides character-by-character communication that looks, to the MICRO/PDP-11, like a terminal (Figure 5-27).

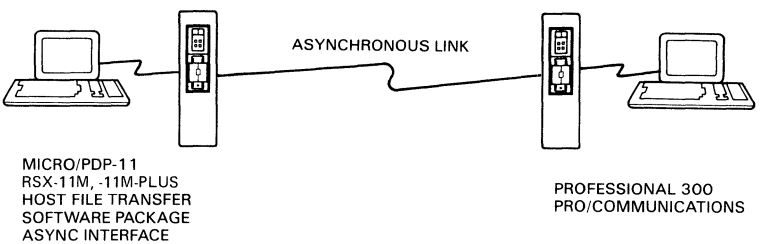


Figure 5-25 File Transfer between MICRO/PDP-11 and Professional 300

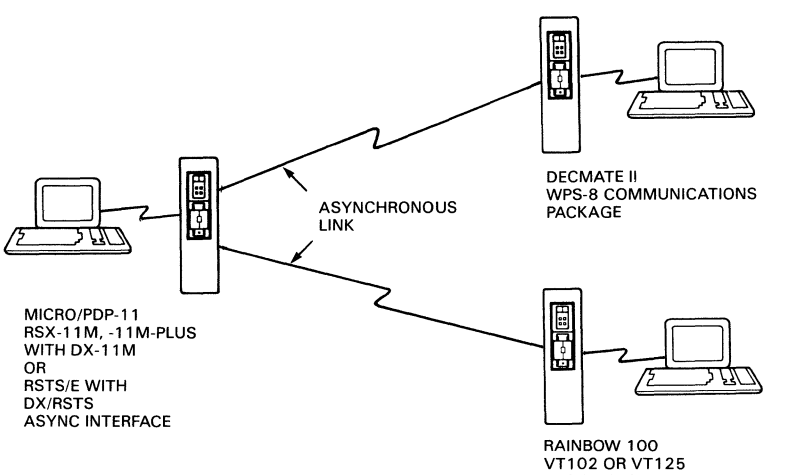


Figure 5-26 File Transfer between MICRO/PDP-11 and DECmate II; between MICRO/PDP-11 and Rainbow 100

Professional 300 with PRO/Communications software can emulate an alphanumeric terminal (VT102) or, with the extended bitmap module, a graphic terminal (VT125).

DECmate II with the WPS-8 package emulates an alphanumeric terminal (VT100 or VT52).

Rainbow 100 emulates an alphanumeric terminal with the VT102 Communications Package, and a graphic terminal with the VT125 Communications Package.

Internet

The MICRO/PDP-11 can communicate with another vendor's equipment through software that emulates a protocol supported by that vendor. Note that, although the name of the protocol may correspond to a specific device made by another manufacturer, the MICRO/PDP-11 emulates only the communications protocol used by that device and not the functionality of the device itself.

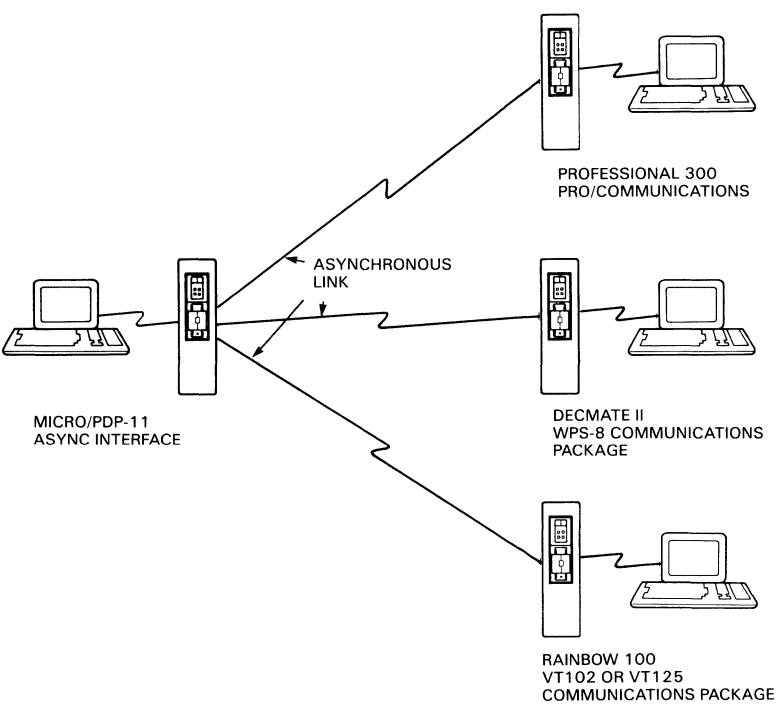


Figure 5-27 Personal Computer Terminal Emulation

The link is a synchronous line. Table 5-2 describes which of the MICRO/PDP-11's synchronous interface modules are supported by Internet software.

IBM 2780 and 3780 remote batch terminals can be emulated under the RT-11, RSX-11M, and RSX-11M-PLUS operating systems.

- RT-11 2780/3780 Protocol Emulator
- RSX-11 2780/3780 Emulator

An IBM 3277 display unit connected to an IBM 3271 controller can be emulated under RSX-11M or RSX-11M-PLUS.

- RSX-11/3271 Protocol Emulator

CDC's 200 UT Mode 4A Communications Protocol can be emulated under RSX-11M.

- MUX200/RSX-IAS Multiterminal Emulator

MICRO/PDP-11 nodes in an Ethernet can also communicate with IBM systems via an SNA Gateway—an Ethernet node solely responsible for interfacing to an IBM system using IBM's System Network Architecture (Figure 5-28).

Packetnet System Interface

Public packet-switched networks can be an attractive alternative to leased or dial-up telephone lines for long-distance communication.

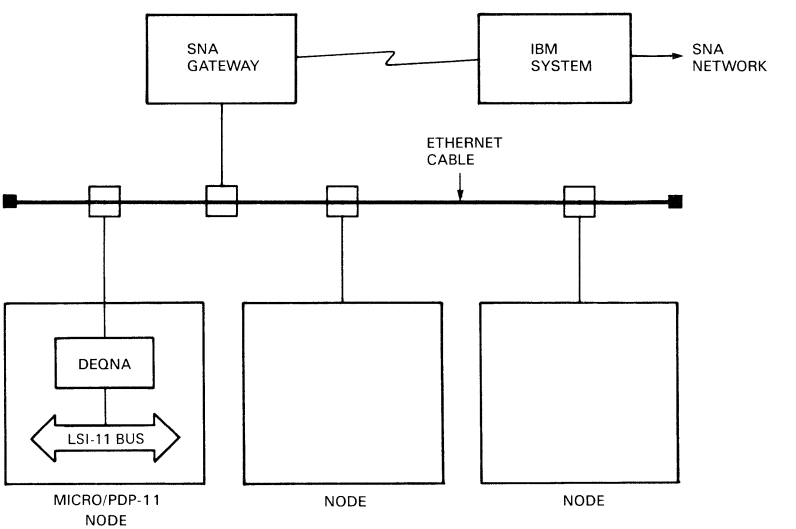


Figure 5-28 SNA Gateway

Subscribers are permanently attached, as with a leased line, but the charge is based on the volume of data transmitted rather than the fixed charge for a leased line. Access, speed, and reliability are apt to be better than that provided by a dial-up line.

The network compensates for differences in transmission speeds between nodes and may offer services in addition to communication.

MICRO/PDP-11 systems running RSX-11M or RSX-11M-PLUS can connect to public packet-switching networks conforming to CCITT recommendation X.25.

- RSX-11 PSI/M Packetnet System Interface
- RSX-11 PSI/M-PLUS Packetnet System Interface

The link to the network is a synchronous line, using the DPV11-DP interface.

The Packetnet System Interface (PSI) software can coexist with, or operate as a layered product under, DECnet software. This allows DECnet facilities to be used between nodes connected via the packet-switched network as well as via leased or dial-up lines. PSI makes communication possible with any other system (DIGITAL or non-DIGITAL) connected to the packet-switched network.

PSI software supports task-to-task communication and remote terminal access to the MICRO/PDP-11 system.

RSX-11 PSI/M and RSX-11 PSI/M-PLUS have been certified and are warranted on Transpac (France), Datex-P (Germany), PSS (United Kingdom) and Telenet (U.S.A.).

CHAPTER 6
ACCESSORIES

DIGITAL's support of your system requirements doesn't stop with the system itself. DIGITAL's Accessories and Supplies Group has the extras you may need to get the most out of your system. And the *DECdirect Catalog* makes getting what you need easy.

MAGNETIC MEDIA

The data you store on your magnetic media—disk cartridges, diskettes—is important to you, and the reliability of its storage is only as good as the media itself. Whether you use the RX50 diskette or the RL02 removable cartridge disk, you can rely on the quality of DIGITAL's magnetic media.

Binders, boxes, suspension files, desk stands and mailers are available for your RX50 diskettes to protect their contents and organize your files (Figure 6-1). Accessories for simplifying media storage are also available for the RL02 cartridges. For complete on-site media storage protection, you can even get media fire safes that protect your media against theft and fire.

FURNITURE

MICRO/PDP-11 has been designed to fit your environment. DIGITAL's furniture is designed to suit the system users. VT100 Modular Furniture can be adjusted to fit the operator, minimizing neck and back strain. There is a split-top table for terminals in the VT100 family such as the VT102 and VT125. The table can be matched with an adjustable-height table, a fixed-height table, a mobile file unit for diskette or paper files, and a set of connector panels to create a variety of workstation configurations. The light oak, nonglare top and charcoal-brown base are compatible with office environments.

For rugged environments, you may want to take advantage of the steel-and-laminate construction of the workstation furniture.

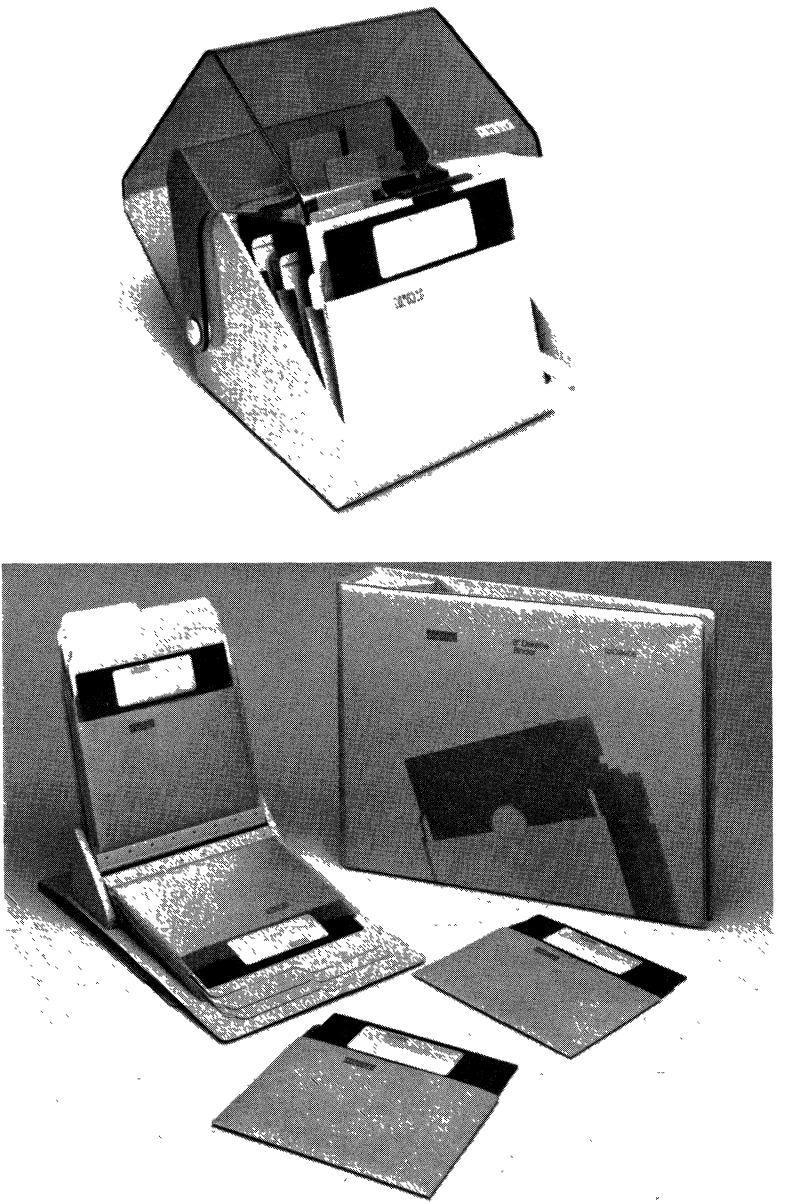


Figure 6-1 Diskette Storage Units

DOCUMENTATION

DIGITAL's famous handbooks are available as well as individual manuals or documentation sets for PDP-11 software and hardware products. Selected documents relating to the MICRO/PDP-11 are listed in Appendix K. For a full listing, see the latest version of the Documentation Products Directory.

PRINTER SUPPLIES

For each of the tabletop printers, there is an optional stand and dust cover. LQP02 and LA100 owners may want to install an acoustic cover to minimize noise in a quiet environment, a paper tray to hold the printed paper, or a tractor for fan-fold paper or forms. LQP02 owners can also choose from a wide selection of snap-in print wheels; the selection includes various typefaces, national character sets, and symbol fonts. LA100 owners can make a change of font by purchasing an add-in font in one of a variety of typefaces.

Nylon and film ribbons for each of the printing terminals are also available.

VIDEO TERMINAL ACCESSORIES

There are several accessories available for users of VT100-family video terminals. The tilt/swivel base adjusts the display to the position most comfortable for a particular operator (Figure 6-2). Antiglare filters are available in several colors for both high ambient light environments and low ambient light environments.

The keyboard can be adapted to specific applications with flexible plastic overlays or with installation of new, customized keycaps.

MODEMS

DIGITAL's DF03 modems connect directly to a modular telephone jack and provide 1200 bps (synchronous or asynchronous) and 300 bps (asynchronous) communication over dial-up telephone lines; they are compatible with Bell 212A and 103J data sets (Figure 6-3). DF03 can be used with any standard telephone for manual call origination, and a DF03 model is available with serial asynchronous auto-dial capability that can be controlled by a computer system or terminal connected to the EIA port.

The same functionality is also available in a single-module modem that mounts in a 10.5-inch-high rack-mountable subsystem, capable of holding up to 12 modems. These modems are available with or without auto-dial (Figure 6-4).

DF02 is a 300 bps modem that connects to the dial-up phone network in the same way as the DF03; DF02 also comes in a variation with auto-dial.

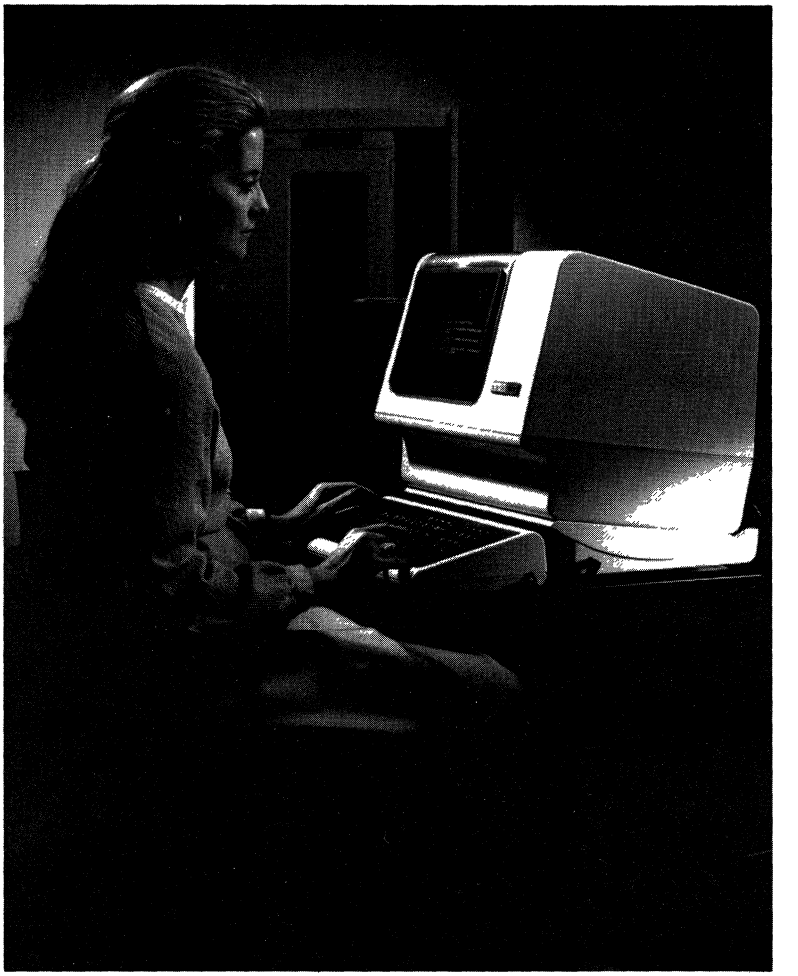


Figure 6-2 VT100 Tilt/Swivel Base

CABINETS

Standard 40-inch-high and 60-inch-high cabinets are available in a variety of configurations: top-load mass storage (for RL02), general-purpose standalone, and expansion. Cabinets can be delivered either fully-assembled or can be ordered in parts (Figure 6-5).

Figure 6-3 DF03

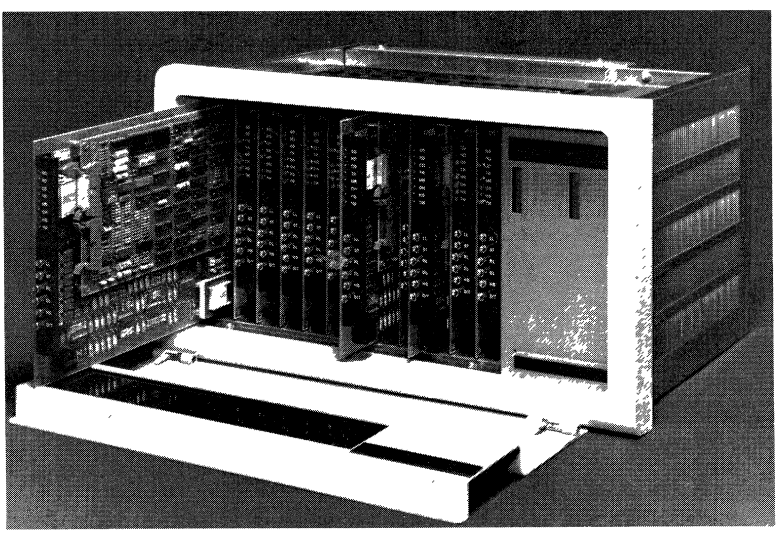


Figure 6-4 Rack-Mount Modem Subsystem

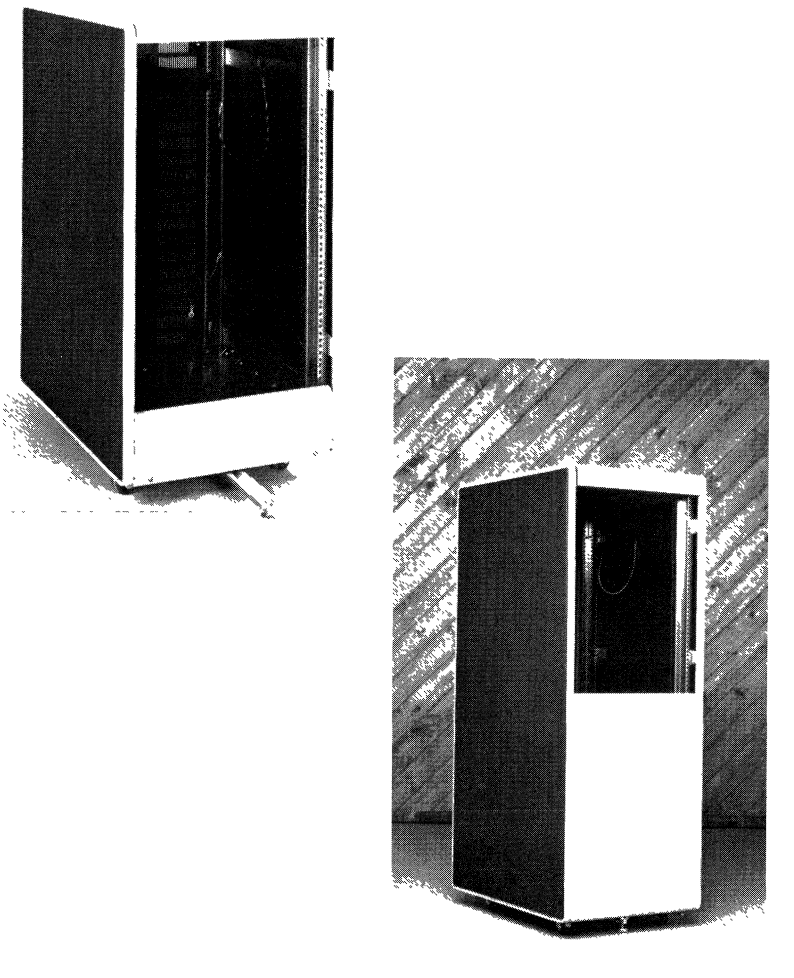


Figure 6-5 Forty-inch and 60-inch Cabinets

CABLES

Cables for use with MICRO/PDP-11 interfaces are available: EIA, EIA null modem, EIA with modem control, Ethernet, DDCMP integral modem, etc. Refer to the configuration details for each interface option to determine which cable is right for your application. If you want to make your own cables, you can get the connectors from DIGITAL.

CHAPTER 7
MICRO/PDP-11 ARCHITECTURE

MICRO/PDP-11 is the most recent implementation of the PDP-11 architecture.

The word "architecture" refers to the system characteristics that are important to a programmer at the assembly language level—memory organization and addressing, registers, data types, instructions, and special features.

With modern system software—operating systems, high-level languages, application programs—the system architecture may seem remote, unimportant, and irrelevant. However, whether or not you interface with the system at the assembly language level, the architecture means something to you.

- An enduring architecture, like the PDP-11, acquires a library of software and an inventory of programmer experience.
- System software can be more efficient when it is written for a powerful, efficient, understandable architecture like the PDP-11.
- An architecture implemented as microprocessor chips, board-level processors, single-board computers, and systems from under $5,000 to over $200,000 (like the PDP-11) gives you a device to fit your needs.
- The powerful VAX-11 architecture is closely related to the PDP-11 architecture. As a PDP-11 user, you have a growth path.

There have been over a dozen different implementations of the PDP-11 architecture. Each has been different from the others in its physical realization. Some have used magnetic core memory, others have used semiconductor memory; Some have used the UNIBUS, others have used the LSI-11 bus; some have used fast logic and parallel operations, others have used straightforward and less costly designs. But all have looked like a PDP-11 to the programmer. The section

"KDF11-B CPU Module" on page 25 describes MICRO/PDP-11 as an implementation of the PDP-11 architecture.

Like most contemporary computers, the PDP-11 stores its programs and data in memory. This chapter discusses the key attributes of the architecture in the following categories:

- How main *memory* is organized
- *Registers* available to the programmer
- The way *data* is represented in memory
- The *instructions* available to manipulate data and control the system
- *Special features*.

MEMORY

The smallest addressable unit in PDP-11 memory is a *byte* (8 bits). Two bytes form a 16-bit *word*. Word addresses are always even—a low-order byte at the word address and a high-order byte at the word address plus 1 (Figure 7-1).

PDP-11 instructions form 16-bit addresses, so the *virtual address space* directly addressable by the instruction is 2^{16} bytes, or 65,536 (64K) bytes.

The actual memory capacity in the PDP-11 architecture, the *physical address space*, is 4,096K (4M) bytes. *Memory management* translates 16-bit virtual addresses into the 22-bit physical addresses needed to address this much memory; memory management is a combination of hardware and system software (Figure 7-2).

Instructions can address memory using either *direct* addressing or *indirect* addressing (also called "deferred"). With direct addressing, the 16-bit address in one of the general registers points directly to the data in memory (Figure 7-3a). With indirect addressing, the 16-bit

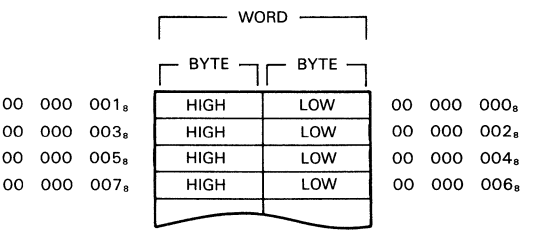


Figure 7-1 Memory Organization

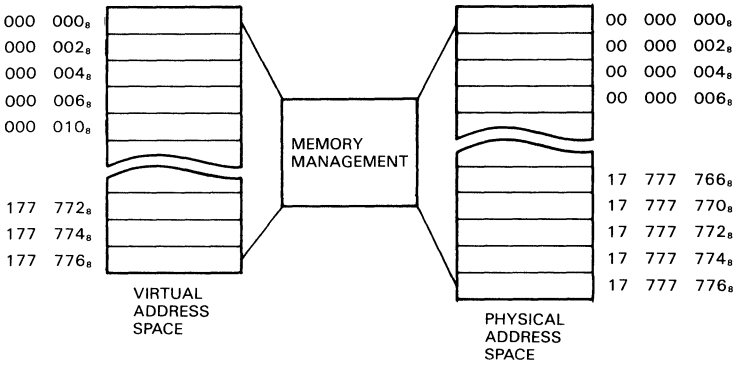


Figure 7-2 Translating 16-Bit Virtual Addresses into 22-Bit Physical Addresses

address in a general-purpose register points to an address stored in memory, which in turn points to the data (Figure 7-3b).

REGISTERS

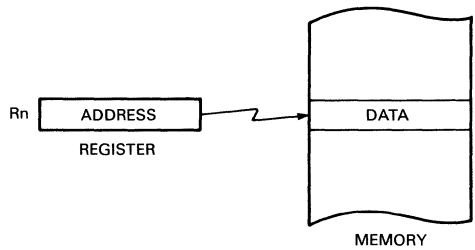
All PDP-11s have six *general registers* (R0–R5), a *processor stack pointer* (R6), and a *program counter* (R7). Each register is 16 bits long (Figure 7-4).

The general registers can be used as operands for arithmetic or logical operations, or in addressing for memory.

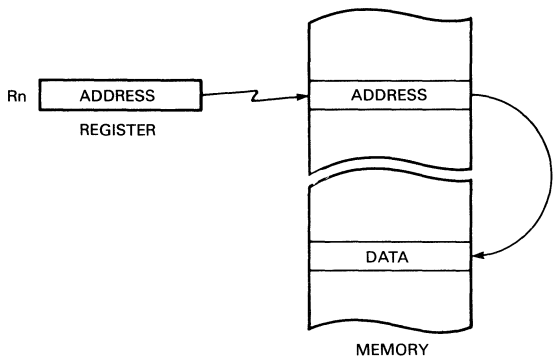
The program counter (R7) contains the address of the next instruction to be executed. Normally, it is only used for addressing and not for arithmetic or logical operations.

When an interrupt or trap occurs, the *processor status word* (PSW) and the program counter (PC) are saved on the processor stack. The processor stack pointer (R6) contains the address of the top of this stack in memory. The PSW and PC contain all the information needed for the processor to resume execution where it left off. The last-in, first-out stack allows orderly processing of interrupts and traps even when the processor is already processing.

PDP-11s equipped with floating-point instructions have six 64-bit *floating point accumulators* (Figure 7-5). These are used to contain operands for floating-point arithmetic and to exchange data with the general registers. A *floating-point status word* (FPSW) describes the current status of floating-point operations.



a. Direct Addressing



b. Indirect or Deferred Addressing

Figure 7-3 Direct and Indirect Addressing

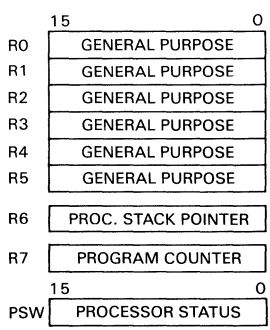


Figure 7-4 Processor Registers

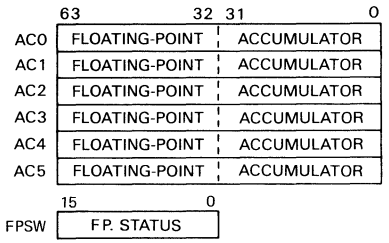


Figure 7-5 Floating-Point Registers

DATA REPRESENTATIONS

Integers

Integers are the most basic numeric data type (Figure 7-6). With software routines, they can emulate the other data types.

Byte integers may be located at even addresses in memory or in the low-order byte of a register (Rn<7:0>). *Word integers* may be at even addresses in memory or in registers. *Long integers* occupy two consecutive registers or two consecutive words in memory.

The least significant bit (LSB) in all integers is bit 0. Signed integers are represented in two's complement form—a negative number is the bit-by-bit complement of the positive number, plus 1. Positive numbers have a most significant bit (MSB) of 0, negative numbers have an MSB of 1.

Floating-Point Numbers

Floating-point data types can represent positive and negative numbers with a much greater absolute value than integer data (as large as 1.7×10^{38}), or with a fractional value (as small as $.29 \times 10^{-38}$).

The value of the number is $(2^K) \times f$, where K is the exponent and f is the fraction. If f is normalized so that $1_2 \leqq f < 1$, there is only one combination of f and K that represents a given number (Figure 7-6).

The *single-precision*, floating-point format allows 23 bits for the fraction (Figure 7-7). However, the MSB of any normalized fraction is 1, so it is not stored explicitly. This 1 is called the *hidden bit*, giving single-precision, floating-point data an effective precision of 24 bits. The *double-precision* format allows 55 bits for the fraction, for 56 bits of effective precision. The *sign bit* is 0 for a positive fraction and 1 for a negative fraction.

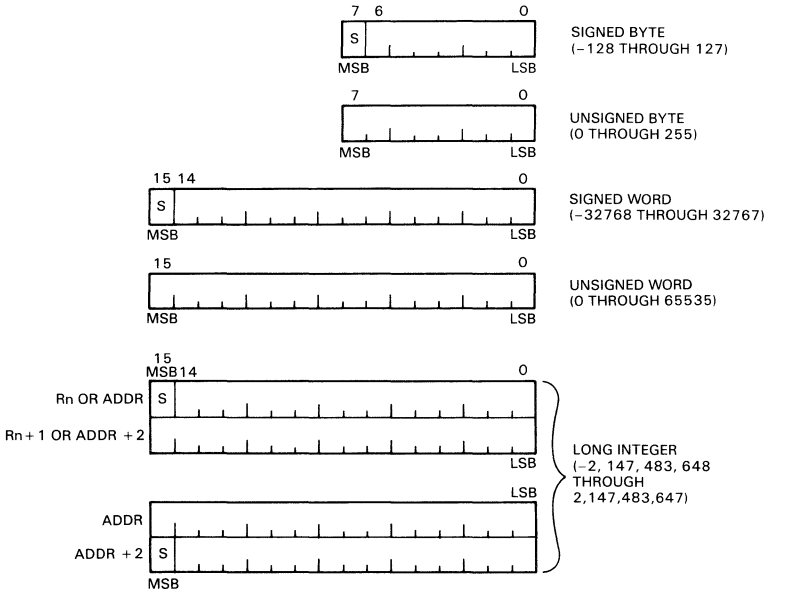


Figure 7-6 Integer Data Types

The value of the exponent K can range from -127 through 127 and is stored as 1 through 255 by adding 128 (excess 128 notation). An exponent of -128, stored as 0, is a special case.

- If all bits of the number are 0, the data represents floating-point zero (a clean or exact zero).
- If the sign bit and the stored exponent are 0 and the fraction is nonzero, the data represents a dirty zero.
- If the sign bit is 1 and the stored exponent is 0, the data represents the undefined variable, also referred to as -0.

Characters and Character Strings

An 8-bit *character* (Figure 7-8) is stored at any addressable byte in memory or in the low-order byte of a general register (Rn<7:0>). Various standards, the most common of which is ASCII, assign an interpretation to some or all of the 256 different codes that can be represented by this data type.

A *character string* is a sequence of up to 65,535 bytes in memory, identified by a character-string descriptor which can be located in two consecutive registers or two consecutive words in memory (Figure 7-

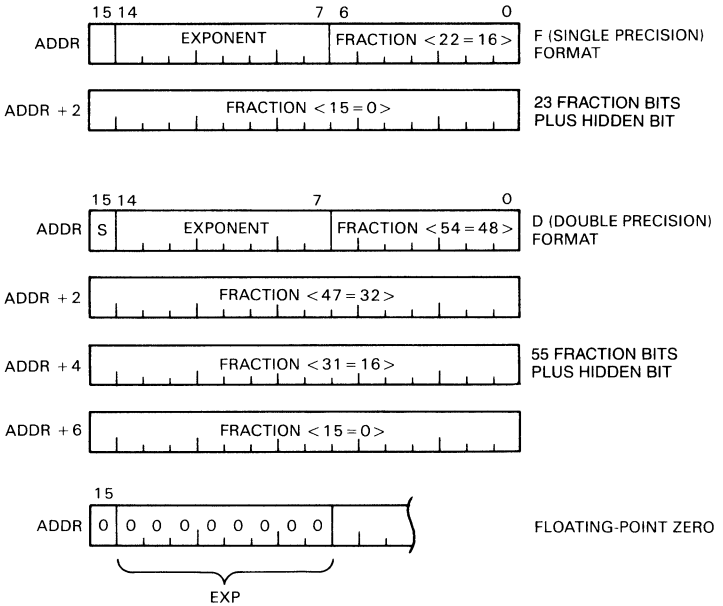


Figure 7-7 Floating-Point Data Types

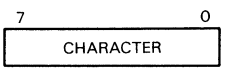


Figure 7-8 Character Data Type

9). The first word is the length of the character string (unsigned integer format), and the second word is the address of the most significant character (MSC). Subsequent characters through the least significant character (LSC) are stored in ascending memory locations.

In some applications, it is helpful to be able to easily specify subsets of the 256 possible character codes. Examples might include the codes in the 7-bit ASCII character set, uppercase letters, numeric characters, or control character codes. The *character set* data format is a convenient way of specifying subsets according to up to eight characteristics. The *character set descriptor*, located in two adjacent registers or two consecutive words in memory, points to a 256-byte table, one byte per character code (Figure 7-10). The table entry corresponding to a particular character is located by interpreting the char-

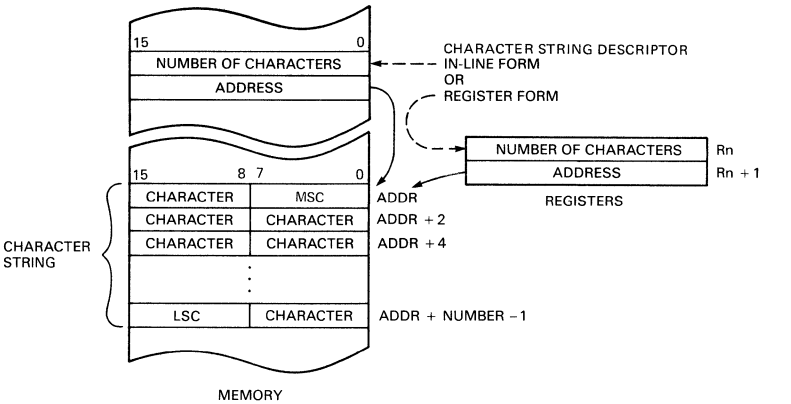


Figure 7-9 Character-String Data Type

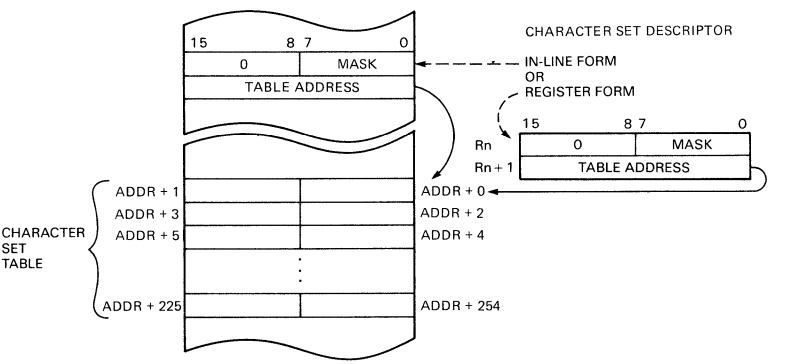


Figure 7-10 Character-Set Data Type

acter as an unsigned 8-bit integer from 0 to 255 and by using the value as an index within the table.

To test whether a character is part of a specified character set, its corresponding table entry is ANDed with the MASK field from the character set descriptor. If the result is nonzero, the character is considered to be part of the character set. For example, if bit 3 of the table entries for all uppercase alphabetic characters is set to 1, and the mask field is 00 000 100, the characters "A," "B," and "C" would be considered part of the character set, while "a," "b," "." and "&" would not.

Decimal Strings

Several kinds of *decimal-string* data formats are used in business applications where their correspondence to COBOL data types, keypunch codes, or printable characters is used. All represent numbers consisting of 0 to 31 decimal digits, with an implied decimal point to the right of the least significant digit (LSD). All are stored in memory as contiguous bytes. Most decimal-string formats code the decimal digits 0–9 as 4-bit nibbles ($0000_2 = 0_{10}$ through $1001_2 = 9_{10}$).

All decimal strings are identified by a decimal-string descriptor located in two consecutive registers or in two consecutive words in memory (Figure 7-11). It specifies the format, points to the address of the most significant digit, and states the length of the string.

There are separate instructions for numeric-string operations and for packed decimal-string operations, and for decimal conversions between the two. The type code in the decimal-string descriptor identifies the exact format (Table 7-1).

Packed decimal strings store two decimal digits per byte: the more significant digit in bits <7:4> and the less significant digit in bits <3:0>. The least significant byte in the string includes a sign nibble in bits <3:0>. Preferred sign codes are generated as the results of a packed decimal operation; alternate sign codes are also properly accepted in the source strings (Figure 7-12).

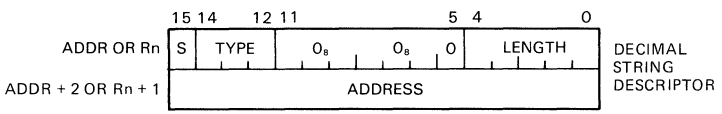


Figure 7-11 Decimal-String Descriptor

Table 7-1 Decimal-String Type Codes

Numeric-Decimal Type Codes	Packed-Decimal Type Codes
0 Signed zoned	0–5 Reserved
1 Unsigned zoned	6 Signed packed
2 Trailing overpunch	7 Unsigned packed
3 Leading overpunch	
4 Trailing separate	
5 Leading separate	
6,7 Reserved	

Packed decimal zero is represented by a zero-length string, stored in one byte (Figure 7-13).

Zoned decimal strings store one decimal digit per byte in bits <3:0> of each byte. The high-order nibble is ignored in arithmetic processing, except in the high-order nibble of the most significant digit of signed zoned decimal strings, which contains a sign code (Figure 7-14).

Although arithmetic processing ignores the high-order nibble by setting it to 0011, the byte corresponds to the printable ASCII character for that decimal digit. Zoned decimal instructions return strings with 0011 in the high-order nibble of all digits, except the sign code nibble of signed zoned strings.

Zoned decimal zero is represented by a descriptor specifying zero length. The string occupies no memory, and the address portion of its description has no meaning.

Overpunch strings are similar to zoned signed strings, except in the way the sign is coded (Figure 7-15). The sign and the most significant digit (in leading overpunch format) or the sign and the least significant digit (in trailing overpunch format) are coded into a single byte corresponding to an ASCII character (Table 7-2).

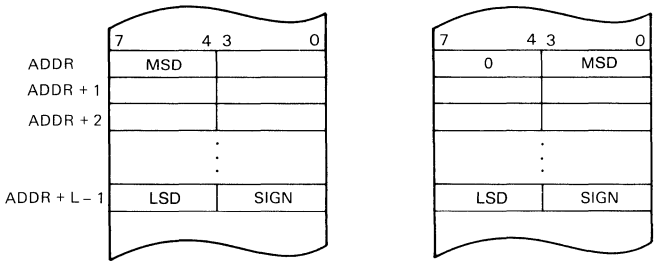
The results of overpunch string operations will always use the preferred codes. Source strings can use either the preferred or alternate codes.

Separate strings are also similar to zoned strings, except in coding the sign (Figure 7-16). Trailing separate strings add a sign byte after the least significant digit. Leading separate strings add a sign byte before the most significant digit. The address in the descriptor points to the MSD, and the length in the descriptor describes the number of digits excluding the sign.

INSTRUCTIONS

PDP-11 instructions are flexible. Each instruction does a lot for the user by making efficient use of program space. The regularity of the instructions makes them easy to understand.

Each instruction consists of an op code that describes its function followed by zero, one, or two operands which represent the data on which the instruction operates.Operands used as input to an instruction are called source operands, and those referring to results are called destination operands. The actual op code is a binary number.



SIGN NIBBLE	PREFERRED CODE	ALTERNATE CODES
+	1100	1010,1110,1111
–	1101	1011
UNSIGNED	1111	

Figure 7-12 Packed Decimal Strings

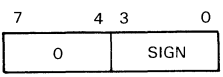
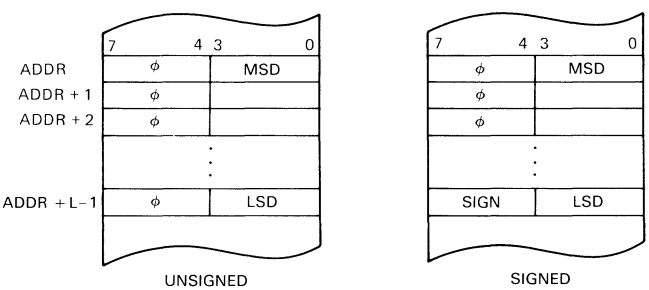
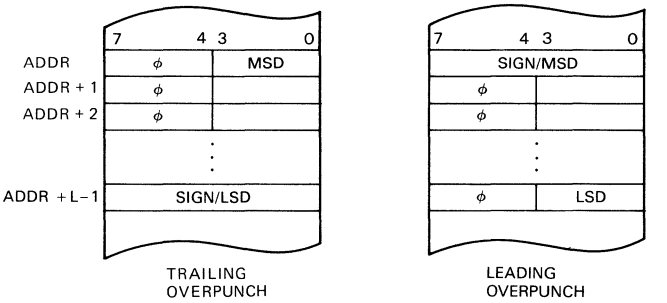


Figure 7-13 Packed Decimal Zero



ϕ = IGNORED IF SOURCE, 0011 IF RESULT
SIGN = 0011 (POSITIVE) OR 0111 (NEGATIVE)

Figure 7-14 Zoned Decimal Strings

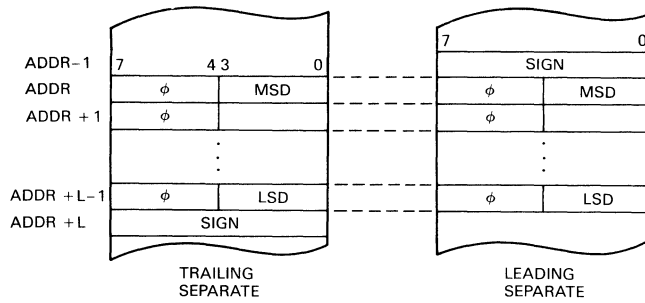


ϕ = IGNORE IF SOURCE, 0011 IF RESULT

Figure 7-15 Overpunch Decimal Strings

Table 7-2 Overpunch Digits

Overpunch Sign/Digit	Preferred Designator (Octal)	Alternate Designator (Octal)
+0	173 ({)	060 (0),133([),077(?)
+1	101 (A)	061 (1)
+2	102 (B)	062 (2)
+3	103 (C)	063 (3)
+4	104 (D)	064 (4)
+5	105 (E)	065 (5)
+6	106 (F)	066 (6)
+7	107 (G)	067 (7)
+8	110 (H)	070 (8)
+9	111 (I)	071 (9)
-0	175 (})	135 (]),041(!),072(:)
-1	112 (J)	
-2	113 (K)	
-3	114 (L)	
-4	115 (M)	
-5	116 (N)	
-6	117 (O)	
-7	120 (P)	
-8	121 (Q)	
-9	122 (R)	



φ = IGNORED IF SOURCE, 0011 IF RESULT

SIGN = 053_8 (ASCII '+') PREFERRED +
040_8 (ASCII ' ') ALTERNATE +
055_8 (ASCII '−') −

Figure 7-16 Separate Decimal Strings

To simplify understanding the instructions, each instruction also has a mnemonic name.

Load, store, and move instructions relocate data among registers, memory, and I/O devices (Table 7-3). Arithmetic instructions perform operations that interpret the data as numeric: add, multiply, negate, etc. (Table 7-4). Shift and rotate instructions reposition data within its original location (Table 7-5). Data conversion instructions translate one data type to another (Table 7-6). Logical instructions perform operations that manipulate bits, compare operands, and search for characters (Table 7-7). Program control instructions redirect the flow of execution (Table 7-8). Miscellaneous includes the remaining instructions (Table 7-9).

Most instructions, other than floating-point and string data instructions, use one of three basic formats: single operand, double operand, and branch (Figure 7-17). Refer to Appendix F for more detailed descriptions of PDP-11 instructions.

The single operand for instructions such as CLR (clear) and NEG (negate) is specified by the destination-address field, and the result is left in the same location.

Instructions such as ADD and SUBtract use two operands specified by the source address and the destination address as input. These instructions leave the result at the destination address.

Table 7-3 Load, Store and Move Instructions

Mnemonic	Function
CLR/CLRB	Clear word/byte operand
MOV/MOVB	Move word/byte operand
MFPT	Move from processor type
Floating-Point Data	
LDF/LDD	Load single/double precision word
STF/STD	Store single/double precision word
CLRF/CLRD	Clear single/double precision word
LDEXP	Load exponent
STEXP	Store exponent
STFPS	Store floating-point program status word
STST	Store floating-point status
String Data	
MOVC(I)	Move character string
MOVRC(I)	Move reverse-justified character string
MOVTC(I)	Move translated character string
L2D	Load two string descriptors
L3D	Load three string descriptors

Table 7-4 Arithmetic Instructions

Mnemonic	Function
Integer Data	
ADD	Add
ADC/ADCB	Add carry bit to word/byte
SUB	Subtract word
SBC/SBCB	Subtract carry bit from word/byte
MUL	Multiply
DIV	Divide
INC/INCB	Increment word/byte
DEC/DECB	Decrement word/byte
NEG/NEGB	Negate (take 2's complement) of word/byte
SXT	Sign extend
Floating-Point Data	
ADDF/ADDD	Add single/double precision numbers
SUBF/SUBD	Subtract single/double precision numbers
MULF/MULD	Multiply single/double precision
MODF/MODD	Multiply and separate integer and fraction (single/double precision)
DIVF/DIVD	Divide single/double precision numbers
ABSF/ABSD	Take absolute value of single/double precision number
NEGF/NEGD	Negate single/double precision number

Table 7–4 Arithmetic Instructions (Cont)

Mnemonic	Function
Decimal-String Data	
ADDN(I)	Add numeric decimal strings
ADDP(I)	Add packed decimal strings
SUBN(I)	Subtract numeric decimal strings
SUBP(I)	Subtract packed decimal strings
MULP(I)	Multiple packed decimal strings
DIVP(I)	Divide packed decimal strings

Table 7-5 Shift and Rotate Instructions

Mnemonic	Function
ASH	Arithmetic shift register
ASHC	Arithmetic shift two combined registers
ASL/ASLB	Arithmetic shift word/byte left
ASR/ASRB	Arithmetic shift word/byte right
ROL/ROLB	Rotate register left
ROR/RORB	Rotate register right
SWAB	Swap bytes in word
Decimal-String Data	
ASHN(I)	Arithmetic shift numeric decimal string
ASHP(I)	Arithmetic shift packed decimal string

Table 7-6 Data Conversion Instructions

Mnemonic	Function
LDCID	Load and convert integer → double precision floating
LDCIF	Load and convert integer → single precision floating
CVTLN	Long integer → numeric decimal
CVTLP	Long integer → packed decimal
LDCLF	Load and convert long integer → single precision floating
LDCLD	Load and convert long integer → double precision floating
CVTNL	Numeric decimal → long integer
CVTNP	Numeric decimal →packed decimal
CVTPL	Packed decimal → long integer
CVTPN	Packed decimal → numeric decimal

Table 7–6 Data Conversion Instructions (Cont)

Mnemonic	Function
LDCFD	Load and convert single → double precision floating
STCFD	Store and convert single → double precision floating
STCFI	Store and convert single precision floating → integer
STCFL	Store and convert single precision floating → long integer
LDCDF	Load and convert double → single precision floating
STCDF	Store and convert double → single precision floating
STCDI	Store and convert double precision floating → integer
STCDL	Store and convert double precision floating → long integer

Table 7-7 Logical Instructions

Mnemonic	Function
BIS/BISB	Bit set (word/byte)
SCC	Set all condition code bits
SEC	Set carry condition code
SEN	Set negative condition code
SEV	Set overflow condition code
SEZ	Set zero condition code
BIC/BICB	Bit clear (word/byte)
CCC	Clear all condition code
CLC	Clear carry condition code
CLN	Clear negative condition code
CLV	Clear overflow condition code
CLZ	Clear zero condition code
COM/COMB	Take one's complement of word/byte
CMP/CMPB	Compare word/byte
CMPF/CMPD	Compare single/double precision floating point
CMPC(I)	Compare character strings
CMPN(I)/CMPP(I)	Compare numeric/packed decimal strings
BIT/BITB	Bit test (word/byte)
TST/TSTB	Test word/byte
TSTF/TSTD	Test single/double precision floating point
TSTSET	Test word, set low bit
WRTLCK	Read/lock destination, write/unlock RO
LOCC(I)	Locate character
MATC(I)	Match character string
SCANC(I)	Scan character string
SKPC(I)	Skip character string
SPANC(I)	Span character string
CFCC	Copy floating condition codes
XOR	Exclusive OR word

Table 7-8 Program Control Instructions

Mnemonic	Function
BR	Branch (unconditional)
BCC	Branch if carry bit is clear
BCS	Branch if carry bit is set
BVC	Branch if overflow bit clear
BVS	Branch if overflow bit set
BEQ	Branch if equal to (zero)
BNE	Branch if not equal to (zero)
BGT	Branch if greater than (zero)
BGE	Branch if greater than or equal to (zero)
BLT	Branch if less than (zero)
BLE	Branch if less than or equal to (zero)
BHI	Branch if higher
BHIS	Branch if higher or same
BLO	Branch if lower
BLOS	Branch if lower or same
BMI	Branch if minus
BPL	Branch if plus
JMP	Jump
JSR	Jump to subroutine
RTS	Return from subroutine
MTPI	Move to previous instruction space
MFPI	Move from previous instruction space
MTPD	Move to previous data space
MFPD	Move from previous data space
MTPS	Move to processor status word
MFPS	Move from processor status word
CSM	Call supervisor mode

Condition codes—four bits in the processor status word—show the result of executing instructions. Bit C is set if the preceding instruction generated a carry, as in ADDing two unsigned word integers with a resulting sum greater than 65,535. The N bit is set if the result of the previous operation was negative, and the Z bit is set if the result of the previous operation was zero. The V bit is set if the result of the previous instruction could not be represented in the destination operand.

Normally, instructions are executed one after the other, in sequence, through ascending memory locations. Branch conditions alter this flow based on a test of the condition codes. If the condition of the test is

Table 7-9 Miscellaneous Instructions

Mnemonic	Function
TRAP	Trap
BPT	Breakpoint trap
EMT	Emulator trap
IOT	Input/output trap
RTI	Return from interrupt
RTT	Return from interrupt
MARK	Facilitates stack cleanup
HALT	Halt
WAIT	Wait for interrupt
NOP	No operation
RESET	Reset bus
SETD	Set floating double precision mode
SETF	Set floating single precision mode
SETI	Set floating for integer mode
SETL	Set floating for long integer mode

satisfied, the next instruction executed is not the subsequent instruction in memory but one at a specified location in a range of -256 to +254 bytes relative to the subsequent instruction. The actual difference in words is specified by the *offset*, a signed integer in bits <7:0>.

Source-address and *destination address* fields each use three bits to designate an addressing mode and three bits to specify one of the eight general registers. The addressing mode specifies how the register is to be used to locate the operand.

Register mode (mode 0) uses the contents of general register Rn as the instruction operand (Figure 7-18a). Since memory is not accessed, this is the fastest mode. Byte instructions use the low-order byte (bits <7:0>) of the register.

Register deferred mode (mode 1) interprets the contents of register Rn as an address. The address is used to locate the operand in memory (Figure 7-18b). The mnemonic for mode 1 is @Rn or (Rn).

Autoincrement mode (mode 2) interprets the contents of Rn as the address of the operand in memory and, in addition, increments the contents of Rn by 2 (word instructions) or by 1 (byte instructions) *after* the operand is accessed in memory (Figure 7-18c). This leaves Rn pointing to the next consecutive word or byte and makes stepping through a list of operands easier. Since both R6 (stack pointer)and R7

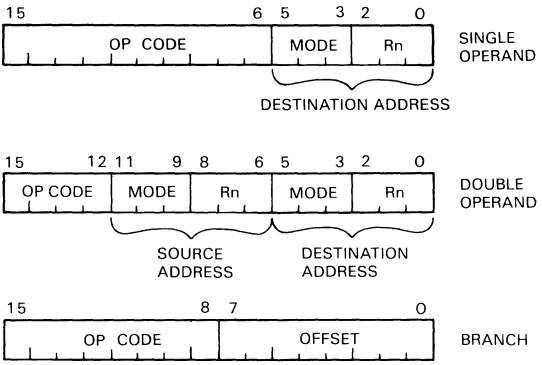


Figure 7-17 Single-Operand, Double-Operand, and Branch Formats

(program counter) normally contain addresses, they are always autoincremented by 2. The mnemonic for mode 2 is (R/n)+.

Autoincrement deferred mode (mode 3) uses the contents of Rn as the address of a word in memory which in turn is used as a pointer to the actual operand (Figure 7-18d). Rn is then incremented 6 × 2. Where mode 2 steps through a list of sequential operands, mode 3 steps through a list of sequential addresses that in turn point to operands stored anywhere in memory. The mnemonic for mode 3 is @(Rn)+.

Autodecrement mode (mode 4) decrements the contents of Rn by 2 (word instructions) or by one (byte instructions) *before* using Rn as the address of an operand in memory (Figure 7-18e). Where mode 2 steps through a list of operands at ascending addresses, mode 4 does the same by descending addresses. The mnemonic is –(Rn).

Autodecrement deferred mode (mode 5) interprets the contents of Rn as the address of a word in memory, which in turn points to the operand (Figure 7-18f). The register is decremented by 2 *before* accessing the address in memory. Where mode 3 steps through a list of addresses in ascending memory order, mode 5 steps through the list in descending memory order. The mnemonic is @–(Rn).

Index mode (mode 6) adds the contents of the word immediately following the instruction to the contents of register Rn, and uses the resulting sum as the address of an operand in memory (Figure 7-18g). This allows you to specify the starting address of a list independently of the offset of an entry in the list. By changing the starting address but not the index, you can move the fifteenth entry in list A to the fifteenth entry in list B; by changing the index but not the starting

address, you can move from the fifteenth entry in list A to the twentieth entry in list A. The starting address can be specified in Rn and the offset in the word following the instruction, or vice versa. The mnemonic is X(Rn).

Index deferred mode (mode 7) adds the word following the instruction to the register Rn in the same way as mode 6, but the resulting sum is used as the address of a word in memory which in turn points to the operand (Figure 7-18h). Where mode 6 accesses operands stored in a list or table, mode 7 uses addresses stored in a list or table to access operands stored anywhere in memory.

Modes 2, 3, 6, and 7 can be particularly useful in conjunction with R7, The program counter. The resulting addressing will be independent of where in memory the instruction is executed.

Each time the processor implicitly uses the program counter to fetch a word from memory, the program counter is automatically incremented by 2 after the fetch is completed.

PC immediate mode (Figure 7-19a) is a special case of mode 2, using the program counter (R7) as Rn. It accesses the word immediately following the instruction and is a fast way to access a constant operand. The mnemonic is #n, where n is the constant.

PC absolute mode (Figure 7-19b) is a special case of mode 3, where an absolute address (e.g., constant regardless of where in memory the instruction is executed) is stored in the word immediately following the instruction. This absolute address is used as a pointer to the operand. The mnemonic is @#A, where A is the address.

PC relative mode (Figure 7-19c) is a special case of mode 6, using the program counter (R7) as the register. The *updated* contents of the program counter (instruction address + 4) are added to the contents of the word immediately following the instruction (the offset) and the sum is used as a pointer to the operand in memory. The offset and the position of the operand relative to the instruction are independent of where they are located in memory, so PC relative mode is helpful in writing position-independent code. The mnemonic is X(PC) or A.

PC relative deferred mode (Figure 7-19d) is a special case of mode 7, using the program counter as the register. The word following the instruction (the offset) is added to the updated program counter (instruction address + 4), and the resulting sum is used as a pointer to an address which in turn contains the address of the operand.

Floating-Point Instructions

Instructions used for floating point come in single- and double-operand versions (Figure 7-20). Bits <5:0> specify the addressing mode

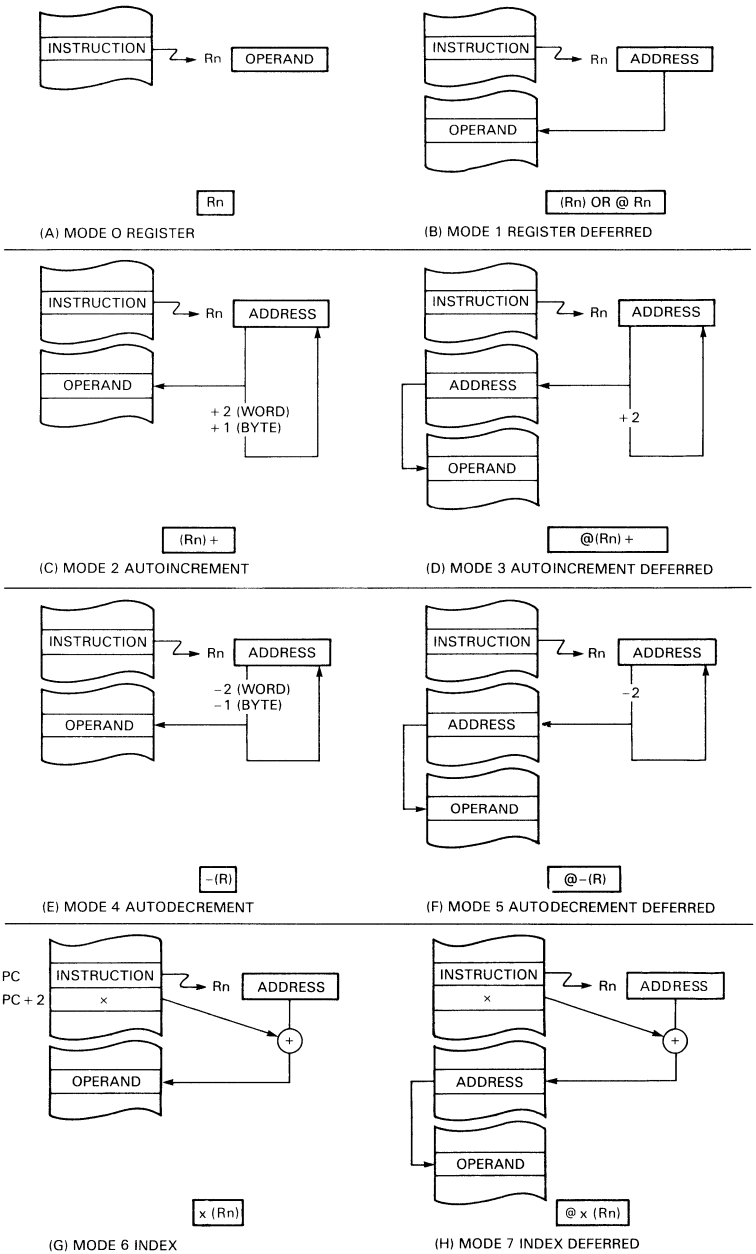


Figure 7-18 Eight Modes

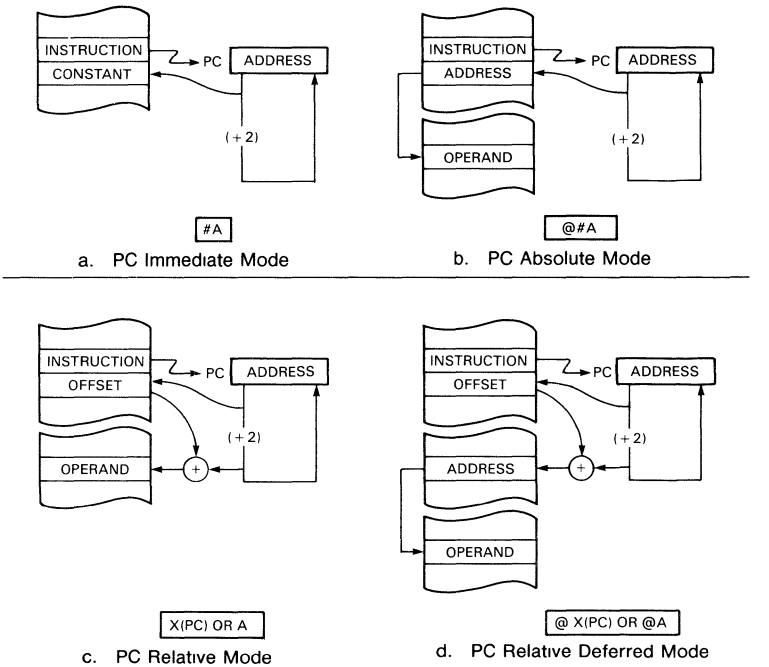


Figure 7-19 PC Mode Addressing

and register of one operand. Bits <7:6> of double-operand, floating-point instructions specify a floating-point accumulator, AC0–AC3, which is used as the other operand.

Depending on the floating-point instruction, bits <5:0> may designate a standard PDP-11 source or destination address (previously described), or a floating-point source or floating-point destination address. In addressing modes 1–7, floating-point source and destination operands are located the same way as a standard PDP-11 source or destination operand. (Note: autoincrement and autodecrement modes change the register contents by 4 for single precision data and by 8 for double precision.)

When a floating-point source or destination specifies addressing mode 0, however, the designated register is interpreted as one of the six floating-point accumulators AC0–AC5.

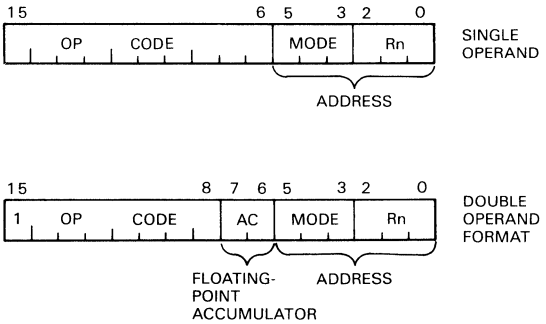


Figure 7-20 Floating-Point Instructions

Character- and Decimal-String Instructions

Instructions that operate on string data—often referred to as the Commercial Instruction Set (CIS)—have completely different formats. Each instruction has a 16-bit op code with fixed, prespecified locations for the operands; the operands include source descriptors, destination descriptors, and other data such as fill characters. Each descriptor comprises two words identifying the location, length, and (for decimal data) the type of the string (see the section "Data Representatives" above for more detail).

Most string instructions come in two variants: register and in-line. In the register form, the string descriptors and nonstring operands (such as fill character) are placed in prespecified general registers (Figure 7-21). The string descriptors point to the actual source and destination strings in memory. Load descriptor instructions facilitate preparation of the general registers for these instructions.

The in-line form of string instructions places string descriptors and other operands in memory immediately following the instruction (Figure 7-22). The string descriptors in turn point to the actual source and destination strings in memory.

SPECIAL FEATURES

Input/Output

The PDP-11 instruction set has no specific input/output instructions. I/O devices have control and data registers that are assigned addresses in the high-order 8KB of the address space. To transfer

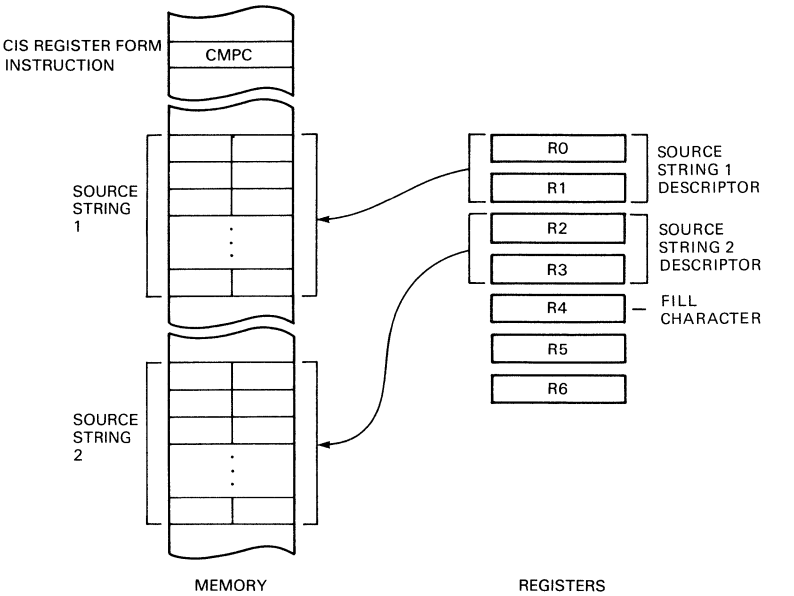


Figure 7-21 CIS Register Form Instruction

data or control information to an I/O device, a program simply writes to the address of the I/O device register (for example, with a MOV instruction). To read data or status information from an I/O device, a program reads from the address of the I/O register.

For example, communication with an asynchronous serial-line interface (connecting the system to a terminal or modem) takes place through four registers (Figure 7-23). The receive control/status register (RCSR) at address 177560_8 can be read to determine whether a character has been received. If it has, the processor can access the character by reading the receive buffer (RBUF) at address 177562_8. The transmit control/status register (XCSR) at address 177564_8 can be read to determine whether the interface is ready to transmit another character. If it is, the processor can pass the character to the interface by writing it into the transmit buffer (XBUF) at address 177566_8. Each of these four buffers is physically part of the serial-line interface and apparently part of the virtual address space.

With program control input/output, described above, the processor tests the device's CSR from time to time to determine whether it is

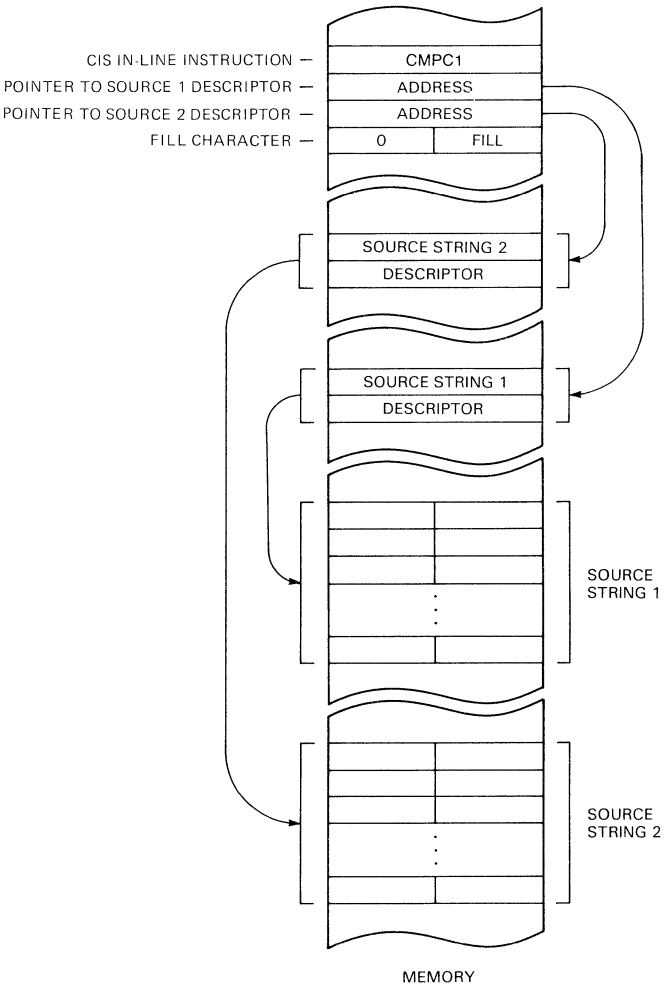


Figure 7-22 CIS In-Line Form Instruction

ready for data transfer. This incurs processor overhead and causes a delay between the time a device is ready for I/O and the time the processor becomes aware and initiates the transfer. The PDP-11 architecture avoids this through interrupt-driven I/O. When the I/O device is ready to transfer data, it sends an interrupt request signal to the processor. When the processor is ready, it returns an interrupt-

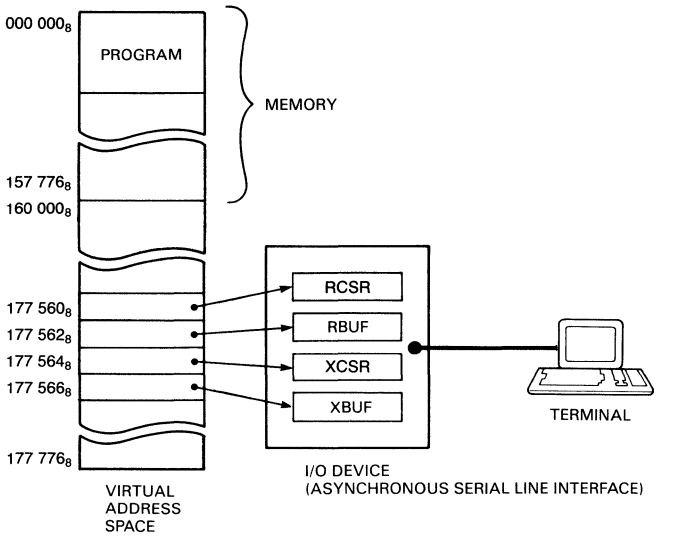


Figure 7-23 Input/Output

acknowledge signal to the device. The I/O device sends its interrupt vector to the processor. The interrupt vector specifies a location in memory containing the address of an interrupt-handling program and a corresponding program status word. The processor saves the system status by pushing the current processor status word and program counter on the R6 program stack and replacing them with the PSW/PC pair identified by the interrupt vector. The effect of all this is a rapid transition from the task in progress to service for the I/O interrupt; when the transfer is complete, the processor can return to the previous task as quickly by restoring the PSW and PC from the top of the R6 program stack.

The most efficient transfers between memory and I/O devices are done with direct-memory-access (DMA) devices and without processor intervention. The I/O device sends a DMA request to the processor; when the I/O device receives a DMA-grant signal back, it is free to complete its transfer. DMA is particularly useful for devices that transfer large amounts of data at high speed, such as disk storage subsystems.

Memory Management

Although the PDP-11's 16-bit address limits the virtual address space seen by a program to 64KB, the architecture supports physical mem-

ory up to 4MB (4096KB) which requires a 22-bit address. Memory management provides the necessary address translation, along with protection mechanisms to ensure the integrity of user data and of system software.

Translating virtual addresses to physical addresses is called relocation. The 64KB virtual address space is divided into 8 pages of 8KB each, and each page is relocated separately into the physical address space.

The high-order three bits <15:13> of a virtual address are interpreted as the active-page field and identify which page the address belongs to (Figure 7-24). The remaining bits <12:0> describe the displacement of the virtual address from the beginning of the page.

The active page field identifies a pair of memory management registers, the page-address register (PAR) and the page-description register (PDR). These are sometimes referred to together as the active-page register (APR). The PAR supplies the beginning address of the page in physical memory space, and the PDR supplies access control, length, and expansion information (Figure 7-25). These registers, along with a set of memory management status registers, are located at addresses in the I/O page.

Under memory management, the top 8KB of the virtual address space can be relocated to the top 8KB of the physical address space for use as the I/O page (Figure 7-26).

Multiple Virtual Address Spaces

The full PDP-11 architecture supports the use of up to six virtual address spaces. The virtual address space is specified by processor operating mode and by selection of instruction or data space (Figure 7-27).

Three processor operating modes are defined in the architecture. Kernel mode allows execution of all instructions. In a multiprogramming environment, the most privileged functions of the operating system—physical I/O operations, resource management, and job scheduling—are implemented in code that runs in kernel mode. The access control provisions of memory management protect these elements from tampering by programs running in less privileged modes.

User mode prohibits the execution of instructions, such as HALT and RESET, that would allow one program in a multiprogramming environment to harm the system as a whole. Each user'svirtual address space permits writing only into its own areas in memory. Supervisor mode has the same level of privilege as user mode and can be useful for programs sharable among users but still requiring protection.

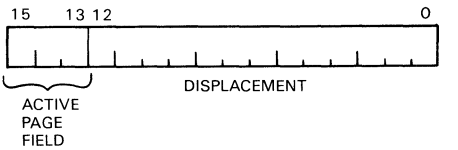


Figure 7-24 16-Bit Virtual Address

The choice of operating mode is determined by a field in the processor status word. Each operating mode has its own set of active-page registers and its own stack pointer (R6). This allows memory management to be done independently for each mode, providing three separate virtual address spaces all relocated into the same physical memory space.

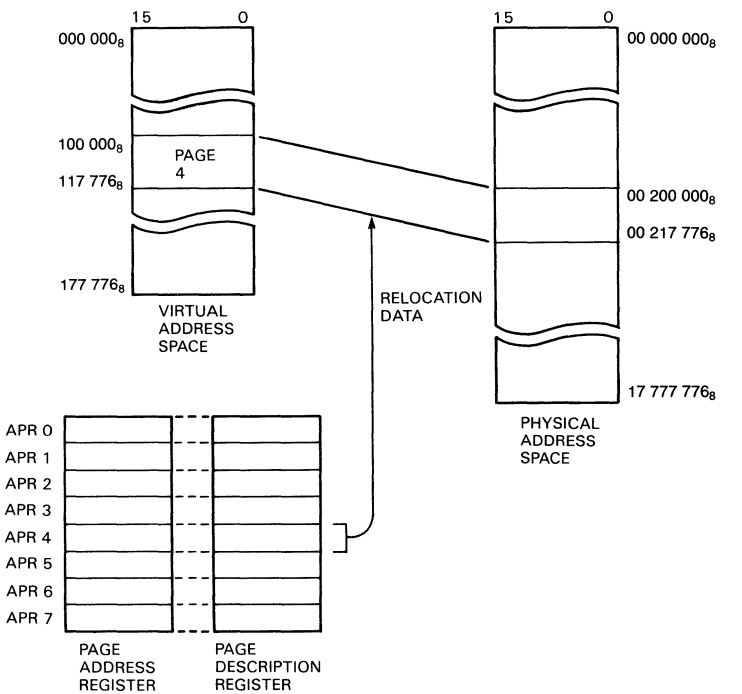


Figure 7-25 Relocation

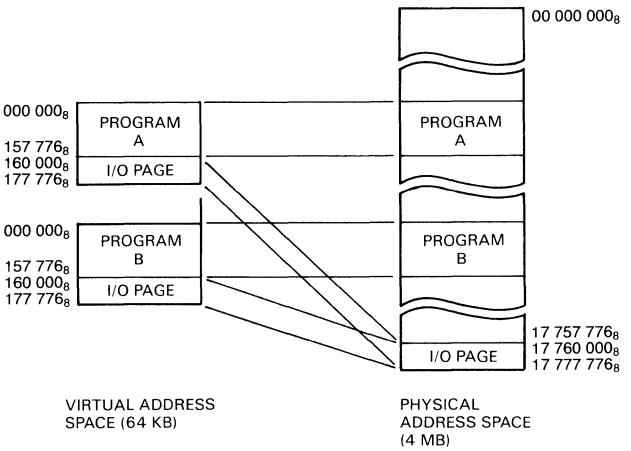


Figure 7-26 Relocating I/O Page

It is possible to map the upper 8KB of the kernel mode virtual address space into the physical address space I/O page, and the upper 8KB of the user and supervisor virtual address spaces into regular memory. This provides a full 64KB virtual address space for user and supervisor programs and limits memory management control and I/O to programs running in the kernel mode, such as operating system components.

An additional doubling of the available virtual address space is possible through the use of instruction space and data space (I & D space). When use of I & D space is enabled through the memory management status registers, an additional set of active-page registers is made available for each of the processor operating modes. Within each mode, there is a 64KB virtual address space for pure instructions (code that does not modify itself as it executes) and another 64KB virtual address space for data.

Bus

MICRO/PDP-11 systems use the LSI-11 bus for communication among the CPU, memory, and peripheral devices (Figure 7-28). The LSI-11 bus is an asynchronous bus: transactions between devices on the bus are limited only by the speed of the devices rather than by a clock as in synchronous buses.

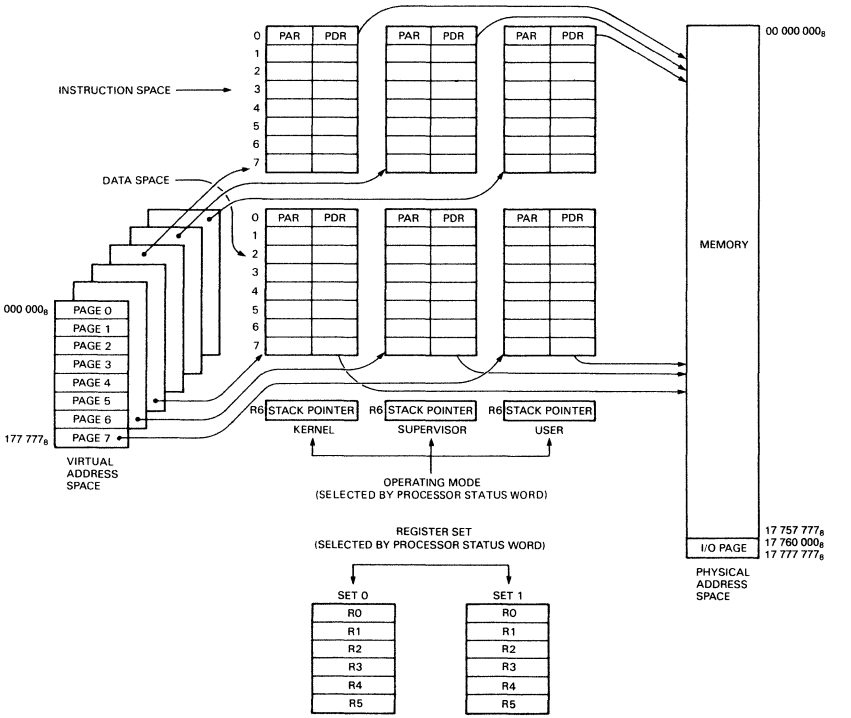


Figure 7-27 Multiple Virtual Address Spaces

There are 22 address lines in the LSI-11 bus, corresponding to the 22 bits of the PDP-11 physical address space. Sixteen of these 22 lines are timeshared for data transfer.

The physical address, generated by the CPU's memory management from a virtual address reference or by the controller of a DMA peripheral, can be put directly on the LSI-11 bus address lines. If the address is in the I/O page, as indicated by a dedicated signal on the LSI-11 bus, the peripheral device with a data or control register at that address will respond. If the address is not in the I/O page, the memory controller responsible for that memory address will respond. This response leads to the bus transaction, examples of which are discussed in Appendix E.

LSI-11 bus transactions take place between a bus master and a bus slave. At any given moment, there can be only one bus master. The CPU arbitrates to determine which device becomes bus master.

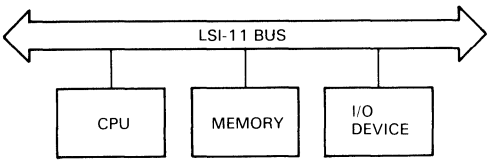


Figure 7-28 LSI-11 Bus

Examples of the master-slave relationship include the CPU (as master) reading from or writing into memory (as slave); a disk controller with direct-memory-access (DMA) capability (as master) transferring data to memory (as slave) without CPU intervention; and the CPU (as master) reading input data from a serial-line interface (as slave).

Once a nonprocessor I/O device has acquired bus mastership, it can transfer data rapidly to or from memory in sequential locations; the only information it requires is the starting address in memory and the length of the transfer. Ordinarily, memory systems require that the memory address of each word of transferred data be specified. This means that the bus master and the memory controller spend 50% of their time transferring addresses and 50% transferring data. Some memory systems are capable of block-mode DMA transfers in which the transfer of a single starting address can be followed by a number of data words; therefore, the proportion of time spent on transferring addresses approaches 0%, and the time spent transferring data approaches 100%.

APPENDIX A
MICRO/PDP-11 SPECIFICATIONS

SIZE

- Rack-mount MICRO/PDP-11:
  13.1 cm high × 47.5 cm wide × 62.5 cm deep
  5.25 inches high × 19 inches wide × 25 inches deep
- Floor-stand MICRO/PDP-11:
  61.25 cm high × 25.4 cm wide × 68.6 cm deep
  24.5 inches high × 10 inches wide × 27 inches deep
- Tabletop MICRO/PDP-11:
  15 cm high × 53.8 cm wide × 68.6 cm deep
  6 inches high × 21.5 inches wide × 27 inches deep

POWER REQUIREMENT

- 120 Vac nominal:
  90–128 Vac, 47–63 Hz
  4.4 A typical, 6.0 A maximum
- 240 Vac nominal:
  180–256 Vac, 47–63 Hz
  2.2 A typical, 4.0 A maximum
- 320 W at rated power supply output
- 7 A fuse

ENVIRONMENT

- 15°–32° Celsius (59°–90° Farenheit), operating
  -40°–60° Celsius (-40°–151° Farenheit), nonoperating
- 20%–80% relative humidity, operating
  10%–95% relative humidity, nonoperating
- 260 W typical, 320 W maximum heat dissipation
  Meets FCC EMI/RFI requirements for a Class A computing device

RX50 DISKETTE DRIVE

- Total capacity 819.2KB formatted
- Dual diskettes, single spindle
- 164 ms average seek
- 100 ms average latency
- 250K bits/sec (31.25K bytes/sec) average transfer rate
- Single-sided 96 tracks/inch preformatted diskettes (RX50K-XX)

RD51 FIXED DISK

- Capacity 9.93MB formatted
- Four surfaces, single spindle
- 5M bits/sec (625K bytes/sec) average transfer rate
- 8.33 ms average latency
- 85 ms average access time (includes setting)

POWER SUPPLY

- 4.5–36 A @ +5 Vdc
  0–7 A @ +12 Vdc
- Total power output: 230 W maximum
- Overvoltage, overcurrent protection
- Thermal Shutdown
- DIGITAL standard ac OK and dc OK signals, and remote controller signal

APPENDIX B
CONFIGURATION GUIDELINES

Some MICRO/PDP-11 options are modules, such as the DZV11 asynchronous multiplexer. Other options, such as the RLV22-AP disk subsystem, include a module that connects an external device (like the RL02 disk drive) to the MICRO/PDP-11. In either case, you will have to answer five questions about your options modules:

1. Are there enough backplane slots available? LSI-11 bus modules come in two sizes: double (13.2 cm × 22.8 cm or 5.2 inches × 8.9 inches).

   The MICRO/PDP-11 backplane has eight slots in all (Figure B-1). The LSI-11 bus is present on the A-B side only of Slots 1–3, and on both the A-B and C-D sides of Slots 4–8.

   - Slot 1 is always occupied by the CPU.
   - Slots 2 and 3 can accept one module—quad or double.
   - Slots 4, 5, 6, 7, and 8 can accept either one quad module or two double modules.

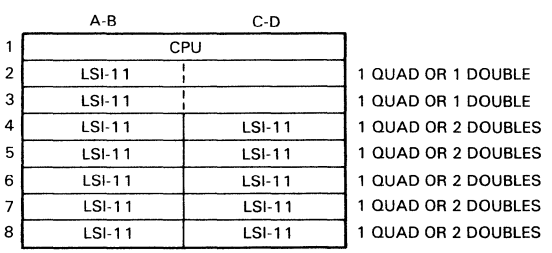


Figure B-1 MICRO/PDP-11 Backplane Diagram

2. How does priority affect module location?

   The interrupt and DMA request lines in the LSI-11 bus follow the pattern shown in Figure B-2.

   - Modules with higher priority should be closer to the CPU along this pattern.
   - If the priorities of your modules require a double module to be located between two quads, you will have to install a bus grant continuity card (G7272) next to the double.

3. Do the modules meet ac bus load limits?

   As specified in Appendix E, a single-backplane system like the MICRO/PDP-11 (with 120 ohm processor termination) can accommodate a total of 35 ac bus loads. The backplane itself contributes 6 ac bus loads.

   Total ac bus loads for all modules in the MICRO/PDP-11 backplane must be no greater than 29.

4. Do the modules and storage meet the power limits?

   The MICRO/PDP-11 power supply provides a maximum of 230 watts of power: up to 36 amps at +5 volts (180 watts) and up to 7 amps at +12 volts (84 watts). Note that it is not possible to draw both the maximum +5 volt and +12 volt currents since that would exceed the 230 watt total limit.

   The MICRO/PDP-11 power supply also supports RD51 and RX50 drives mounted in the system chassis.

   - Total +5 volt current for all modules and disk drives must be 36 amps or less.
   - Total +12 volt current for all modules and disk drives must be 7 amps or less .
   - Total +5 volt power (current @ +5 volts × 5 volts) plus total +12 volt power (current @ +12 volts × 12 volts) must be 230 watts or less.

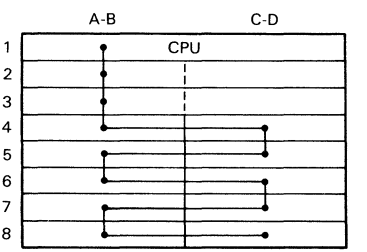


Figure B-2 LSI-11 Bus Priority

5. Do the distribution panel inserts for all modules fit the available space?

   For ease of installation and to ensure minimum EMI/RFI, all external connections to MICRO/PDP-11 modules are made at a distribution panel on the rear of the system (Figure B-3).

   Each module that requires an external connection comes with a distribution panel insert that mounts in one of the apertures (A–F) in the distribution panel.

   The standard configuration provides four 2-inch × 3-inch apertures (A, B, C, and D) and two 1-inch × 4-inch apertures (E and F) .

   By removing the divider between apertures C and D and installing a filler plate, you can create an alternate distribution panel configuration with two 2-inch × 3-inch apertures (A and B) and five 1-inch × 4-inch apertures (E, F, G, H, and J) as shown in Figure B-4.

   The modules in your MICRO/PDP-11 must use *either*:

   - Four 2-inch × 3-inch panel inserts and two 1-inch × 4-inch panel inserts (or fewer)

     or

   - Two 2-inch × 3-inch panel inserts and five 1-inch × 4-inch panel inserts (or fewer).

   The tools you will need to answer these questions are:

   - Table B-1 MICRO/PDP-11 Modules
   - Table B-2 Other Options

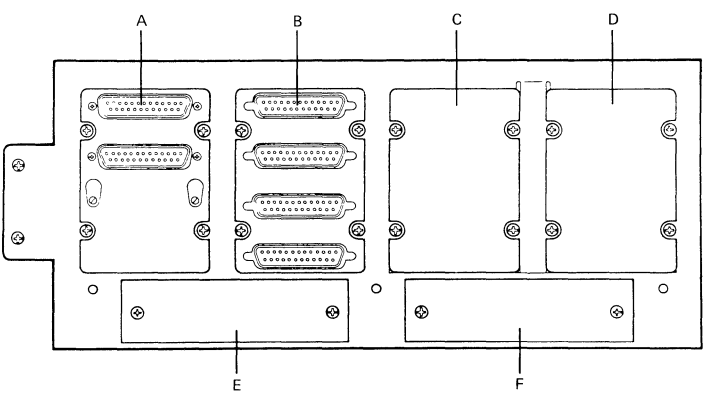


Figure B-3 MICRO/PDP-11 Distribution Panel

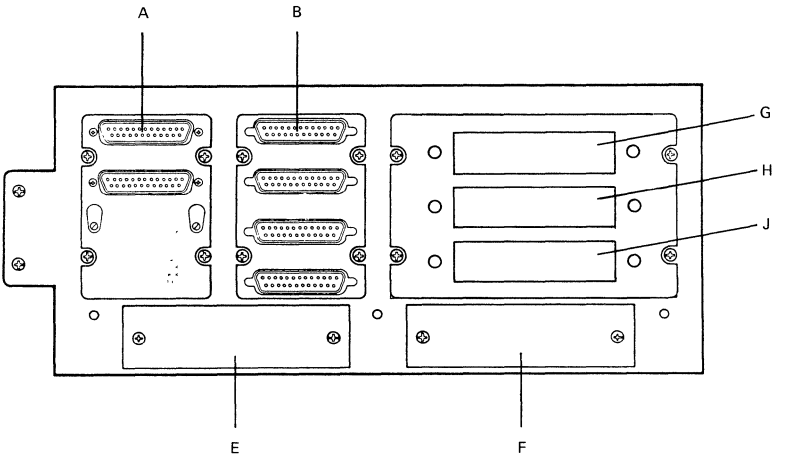


Figure B-4 MICRO/PDP-11 Distribution Panel; Alternate Configuration

- The MICRO/PDP-11 System Chassis Configuration Worksheet (Figure B-5).

Use the configuration worksheet as a guide to positioning modules in the backplane. Base the placement of modules on their priority and size (from the Modules table) and on your answers to questions 1 and 2 above.

Then enter in the configuration worksheet the current requirement at +5 V and +12 V, the number of ac bus loads, and the panel inserts required. Be sure to include from the other-options table the current requirements for RD51 and RX50 if your system uses them.

Starting at the top and working down, subtract the amount of each resource (current, ac loads, panel inserts) *used* by each option from the amount that was *available* before. Write that amount in the new *available* space.

If the *available* number for any resource goes below 0, your system configuration will not work as specified. Call your sales or service representative.

In the panel inserts columns, you should be carrying two *used* and *available* numbers for the two alternate distribution panel configurations. This will help determine which configuration to use.

Note: Partially completed worksheets are included for the standard MICRO/PDP-11 configurations in Figures B-6, B-7, and B-8.

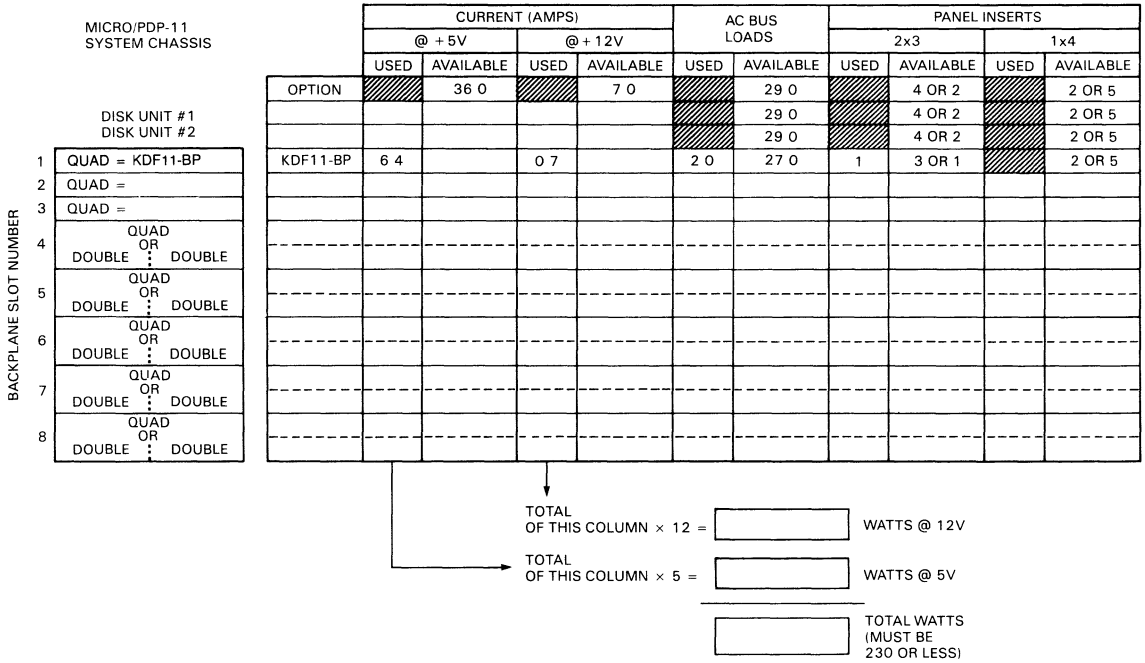


Figure B-5 MICRO/PDP-11 System Chassis Configuration Worksheet

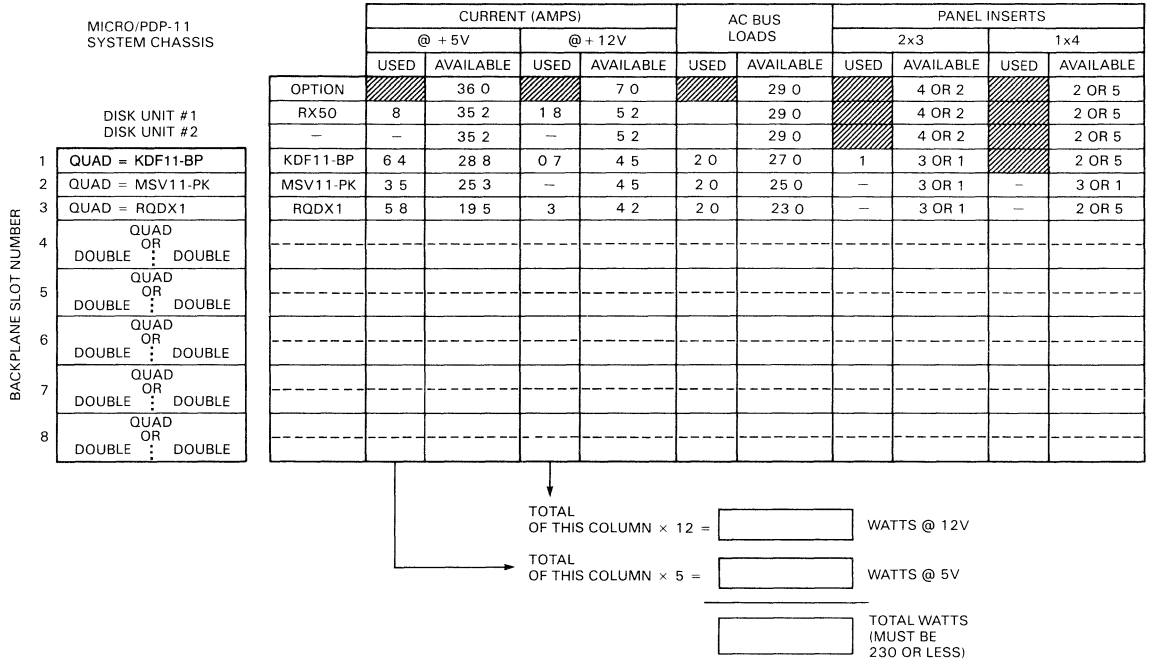


Figure B-6 11A23-R, -F System Chassis Configuration

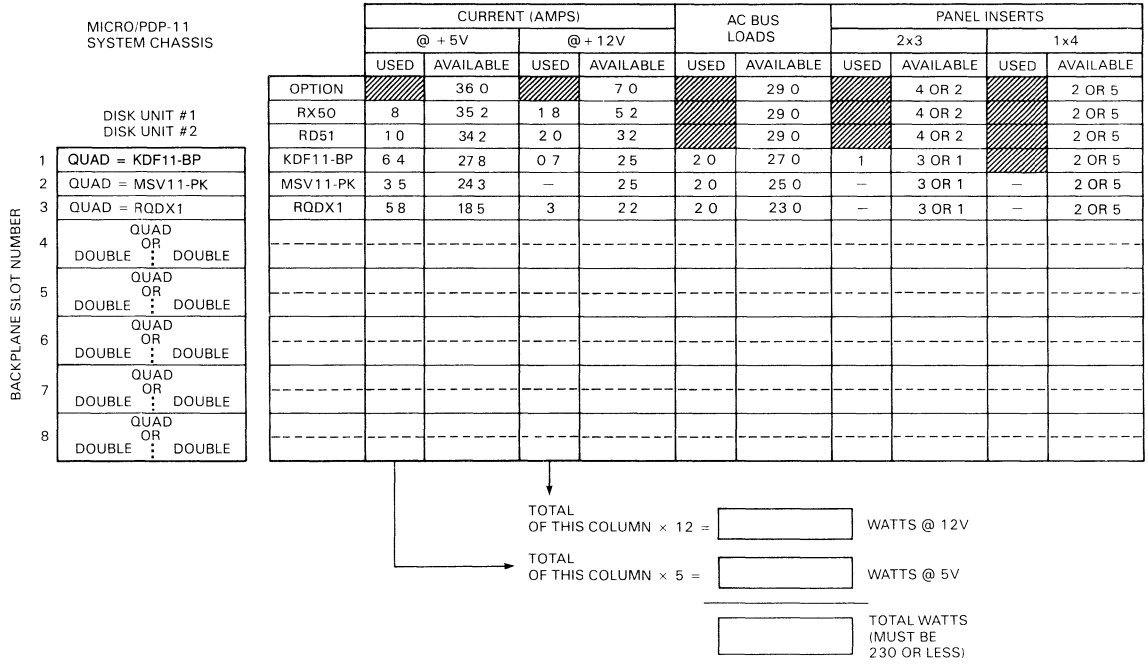


Figure B-7 11C23-R, -F System Chassis Configuration

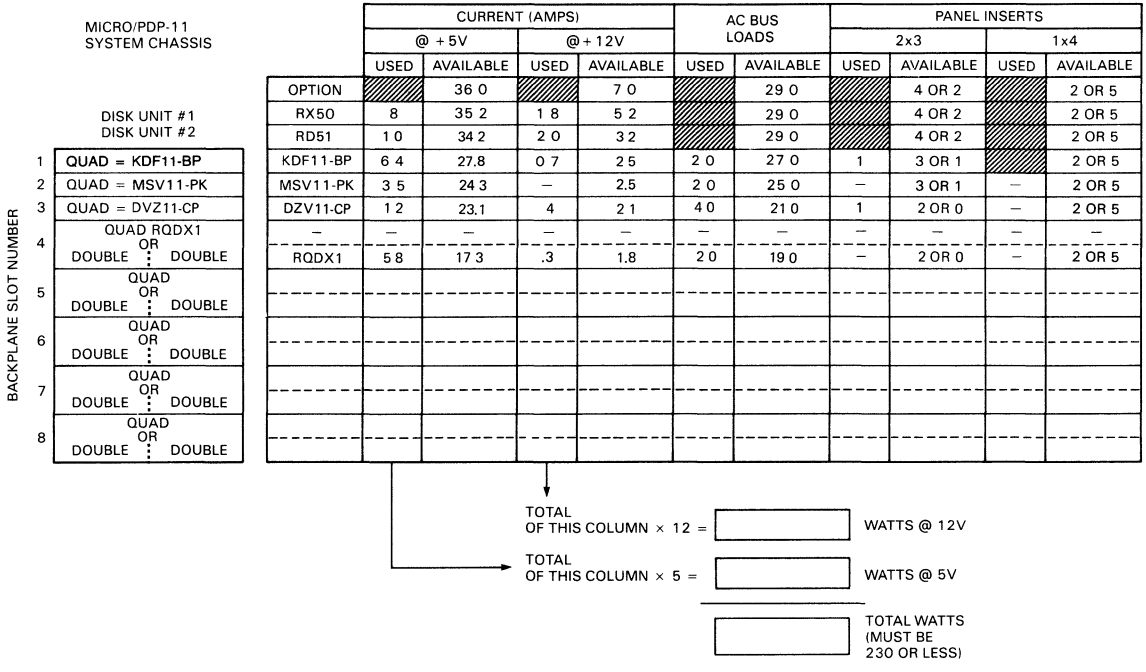


Figure B-8 SX-RA500-FA System Chassis Configuration

Table B-1 MICRO/PDP-11 Modules

Module	Size[1]	Priority[2]	Current (Amps) +5 V	+12 V	Bus Loads ac	Distribution Panel Insert
AAV11-C	D	TBD	2.50	—	0.90	N/A
ADV11-C	D	TBD	2.00	—	1.30	N/A
AXV11-C	D	TBD	2.00	-	1.30	N/A
DLV11-EP	D	M	1.00	0.15	1.60	1 × 4[3]
DLV11-JP	D	M	1.00	0.25	1.00	2 × 3
DMV11-AP	Q	L	3.40	0.38	2.00	1 × 4
DMV11-BP	Q	L	3.40	0.38	2.00	1 × 4
DMV11-CP	Q	L	3.35	0.26	2.00	1 × 4
DMV11-FP	Q	L	3.40	0.38	2.00	1 × 4
DPV11-DP	D	L	1.20	0.30	1.00	1 × 4
DRV11-BP[4]	Q	M	1.90	—	3.30	1 × 4 (2)
DRV11-JP	D	M	1.80	—	2.00	1 × 4 (2)
DRV11-LP	D	M	0.90	—	2.80	1 × 4 (2)
DUV11-DP	Q	L	0.86	0.32	1.00	1 × 4
DZV11-CP	Q	M	1.15	0.40	4.10	2 × 3
FPF11-A	Q	—[5]	5.50	—	—	—
G7272	D	—	—	—	—	—
IBV11-P	D	M	0.80	—	1.77	1 × 4
KDF11-BP	Q	—[6]	6.40	0.70	2.00	2 × 3
KWV11-CP	D	M	2.20	0.01	1.0	1 × 4
LPV11-CP	D	L	0.80	—	1.40	1 × 4
MCV11-DA	D	H	1.20	—	2.00	—
MCV11-DC	D	H	1.20	—	2.00	—
MSV11-LF	D	VH	3.90	—	2.00	—
MSV11-PK	Q	VH	3.45	—	2.00	—
MSV11-PL	Q	VH	3.60	—	2.00	—
RLV12[7]	Q	L	5.00	0.10	2.70	1 × 4
RQDX1	Q	L	5.80	0.03	2.00	TBD
RXV21[4]	D	L	1.80	—	2.00	1 × 4
TSV05-CP	Q	TBD	6.50	—	3.00	1 × 4 (2)

1. Sizes: (D) Double; (Q) Quad
2. Priorities: (VH) Very High; (H) High; (M) Medium; (L) Low; (TBD) to be determined.
3 Can also use 1/4 of DLV11-JP panel insert (70-16436-01).
4. Use with systems up to 256KB.
5. Mounts in slot immediately below CPU. No bus connection.
6. Mounts in top slot.
7. Included in RLV22-AP.

Table B-2 Other MICRO/PDP-11 Options

		Chassis Power Supply Requirements (Amps)	
Option	**Mounting**	**+5 V**	**+12 V**
KEF11-AA	On KDF11-B CPU module	—	—
KEF11-BB	On KDF11-B CPU module	—	—
RD51-A	In MICRO/PDP-11 system chassis	1.00	4.50
RX50-AA	In MICRO/PDP-11 system chassis	0.80	1.80
TU58-EB	Tabletop	—	—

APPENDIX C
MICRO/PDP-11 OPTIONS

These option descriptions are intended to provide a very basic overview of each option's functions, programming interface, and physical interface. For more detailed information, refer to:

- *MICRO/PDP-11 Option Manual* (EK-OLCP5-OD)
- *Microcomputer Interfaces Handbook* (EB-23144-18)
- Specific manuals listed with each option.

Note: Device addresses are located in the I/O page, of physical address space which in the MICRO/PDP-11 is specified with a 22-bit address.

Some LSI-11 bus options appear to support only a 16-bit device address. However, the BBS7 (Bank Select 7) line on the LSI-11 Bus specifies an I/O page reference and has the same effect as specifying the high-order six address bits as ones.

AAV11-C — ANALOG OUTPUT MODULE

Function

Four individually addressable channels of analog output, generated by 12-bit digital-to-analog converters. Output can be unipolar (0 to 10 V) or bipolar (± 10 V). One of the channels has four digital output signals for controlling a user device (such as a CRT). See Figures C-1, C-2, and C-3.

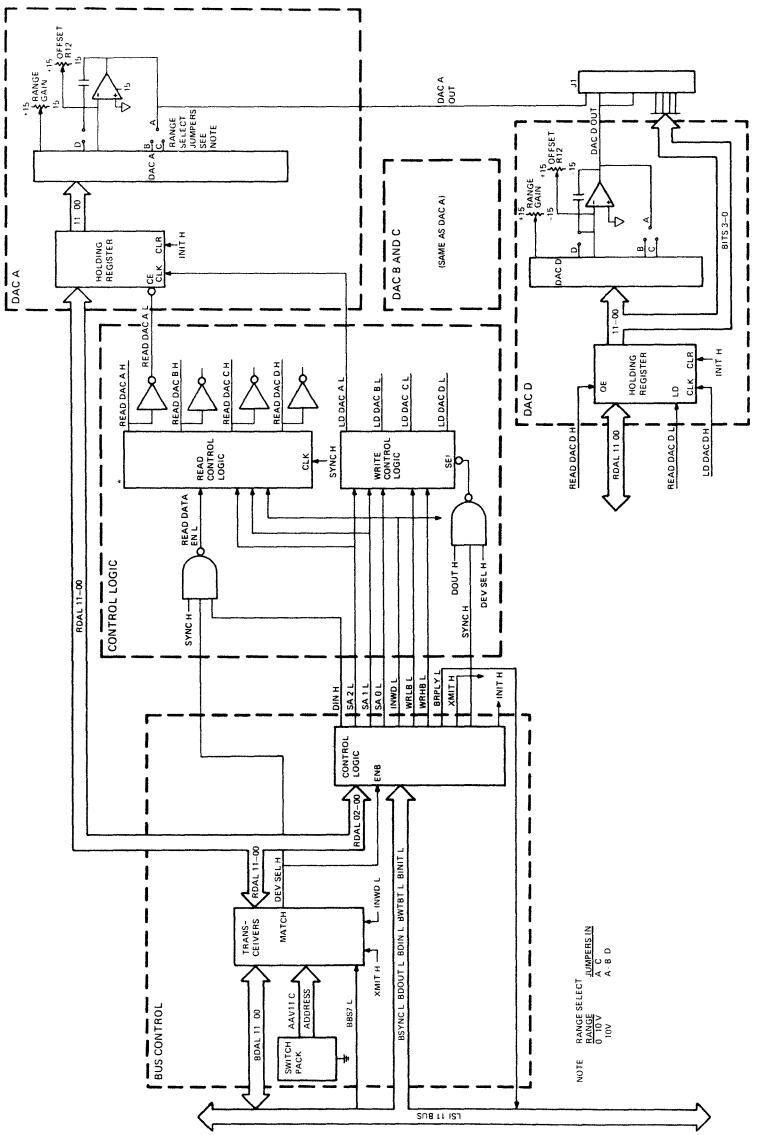


Figure C-1 AAV11 Block Diagram

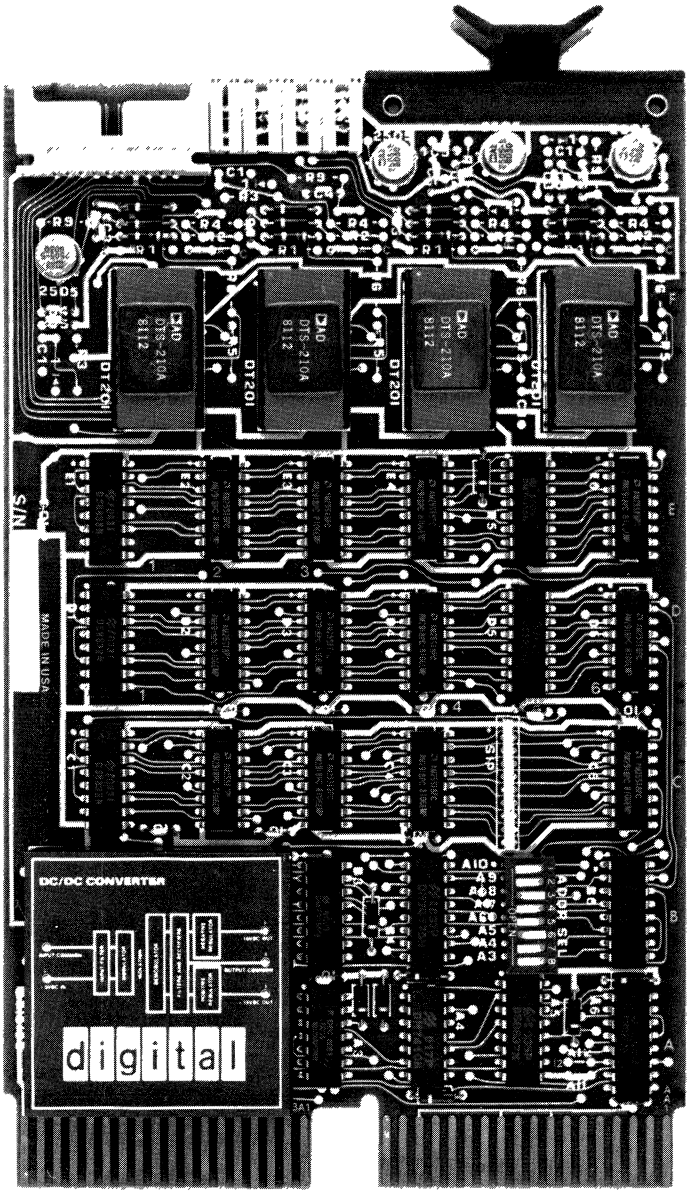


Figure C-2 AAV11

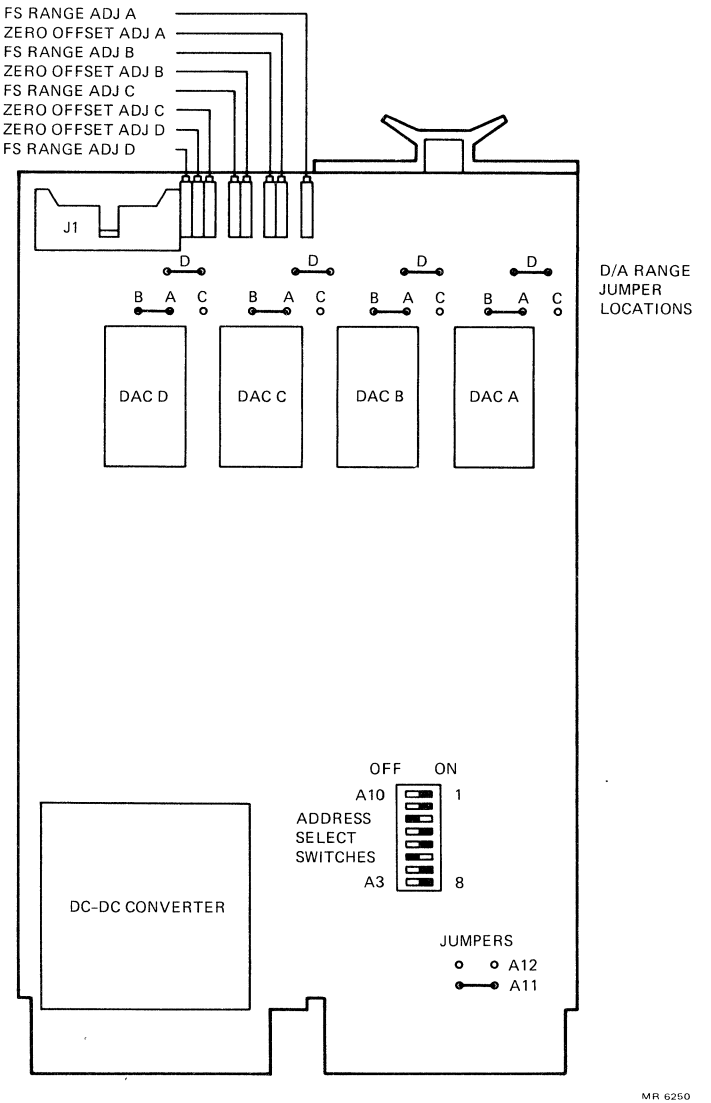


Figure C-3 AAV11 Switch Location

Programming Interfaces

The device base address is set using switch S1 <8:1> and jumpers A11, A12. See Figure C-4.

CSR A	Base Address
CSR B	Base Address + 2
CSR C	Base Address + 4
CSR D	Base Address + 6

The factory-configured base address is 17 770 440_8.

AAV11-C does not require interrupts.

Physical Interface

AAV11-C connects through a 20-pin jack on the module (3M#3421-7020). No distribution panel insert is currently available for this module.

Note: The user is responsible for installation of the AAV11-C module and for ensuring compliance with any applicable EMI/RFI regulations.

Pin 1	D00	2	Digital ground
Pin 3	D01	4	Digital ground
Pin 5	D02	6	Digital ground
Pin 7	D03	8	Digital ground
Pin 9	—	10	—
Pin 11	Analog ground	12	Analog ground
Pin 13	DAC D out	14	Analog ground
Pin 15	DAC C out	16	Analog ground
Pin 17	DAC B out	18	Analog ground
Pin 19	DAC A out	20	Analog ground

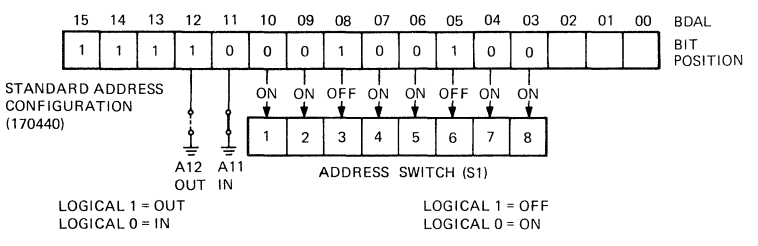


Figure C-4 AAV11 Base Address Selection

Specifications
Double module (A6006)
2.5 A @ +5 Vdc
0.9 ac bus load
1.0 dc bus load

Distribution panel insert: not available

ADV11-C ANALOG INPUT MODULE

Function
Input of 16 single-ended or 8 differential analog signals to the LSI-11 Bus. Unipolar voltages (0–10 V) or bipolar voltages (±10 V) are converted from analog to digital, with an output resolution of 12 bits. Digital data can be represented as binary, offset binary, or two's complement numbers. The gain of each analog input can be multiplied by 1, 2, 4, or 8 under program control.

Analog conversions are started by a program command, an external trigger, or by a real-time clock input. I/O is under program control; an interrupt is generated when the A-to-D conversion is complete. See Figures C-5, C-6, C-7.

Programming Interface
The device base address is set with jumpers A12–A3. See Figure C-8.

CSR (R/W)	Base address	Control/Status register
DBR (R)	Base address +2	Data buffer register

The standard base address is 17 770 400_8.

The vector address is set with jumpers V8–V3. See Figure C-9.

A/D Done	Vector address
Error	Vector address +4

The standard vector address is 400_8.

Physical Interface
ADV11-C connects through a 26-pin jack on the module (3M #3399-7026). No distribution panel insert is currently available for this module.

Note: The user is responsible for installation of the ADV11-C module and for ensuring compliance with any applicable EMI/RFI regulations.

Pin	1	CH0	2	CH8 or Return 0
Pin	3	CH1	4	CH9 or Return 1
Pin	5	CH2	6	CH10 or Return 2

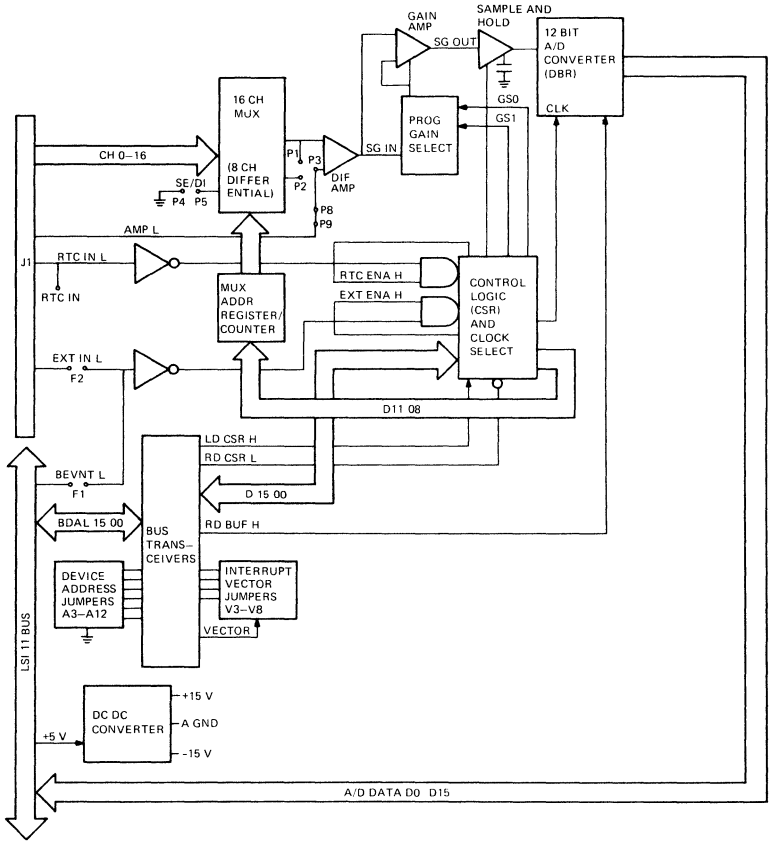


Figure C-5 ADV11 Block Diagram

Pin 7	CH3	8	CH11 or Return 3
Pin 9	CH4	10	CH12 or Return 4
Pin 11	CH5	12	CH13 or Return 5
Pin 13	CH6	14	CH14 or Return 6
Pin 15	CH7	16	CH15 or Return 7

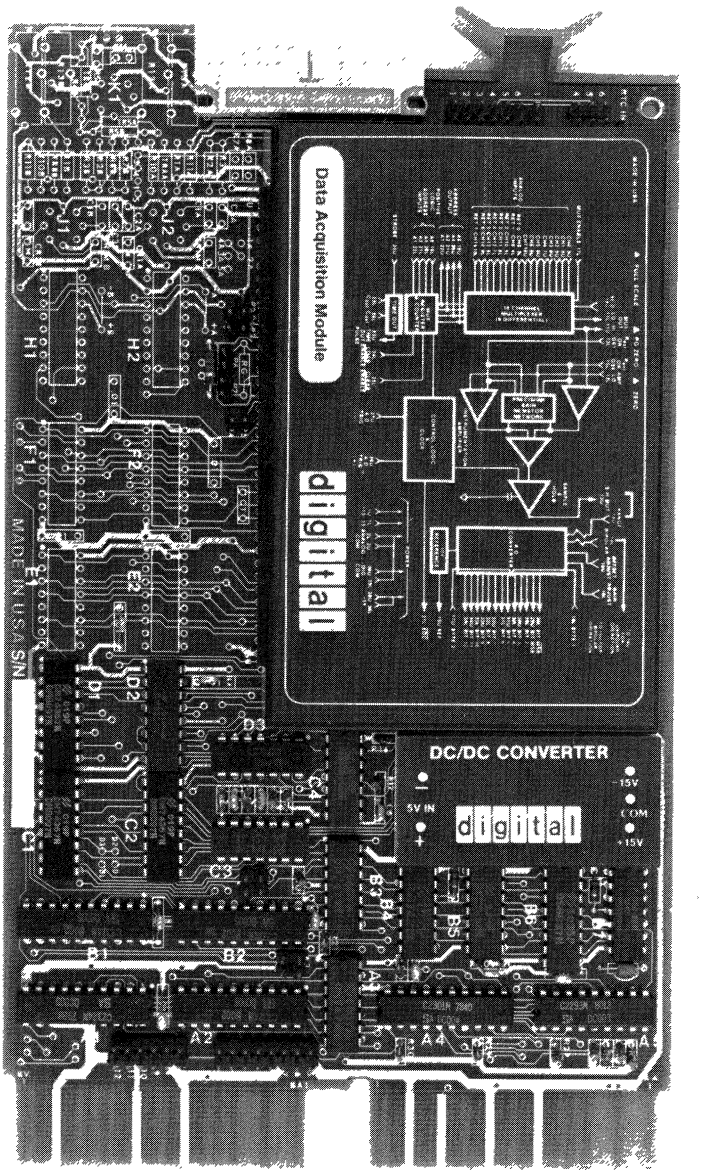


Figure C-6 ADV11

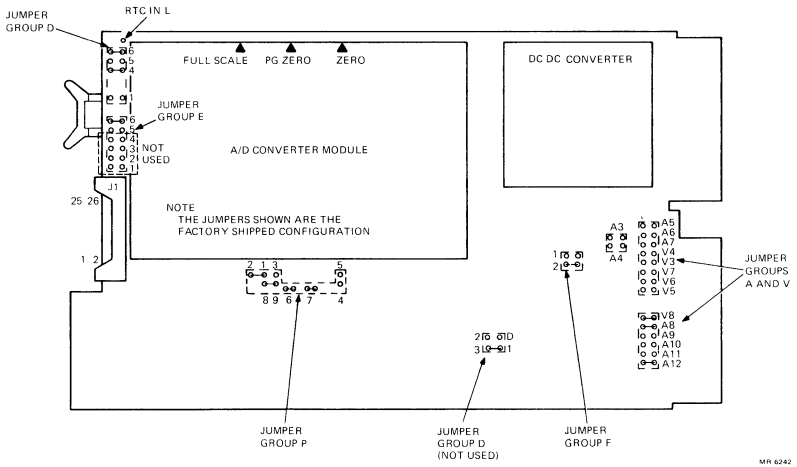


Figure C-7 ADV11 Jumper Locations

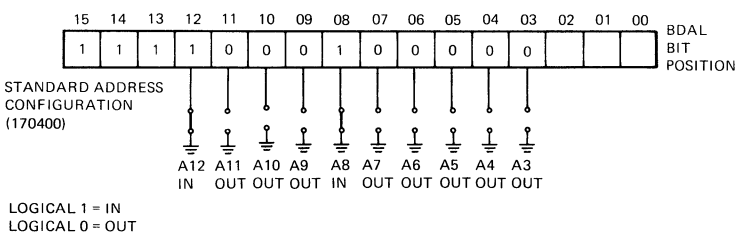


Figure C-8 ADV11 Base Address Selection

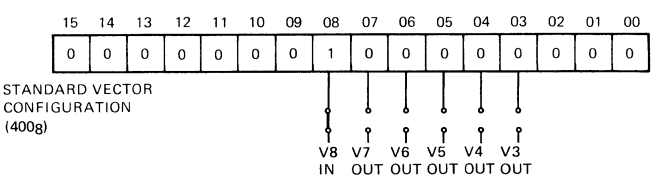


Figure C-9 ADV11 Vector Address Selection

Pin 17	Analog ground	18	AMP
Pin 19	External trigger in	20	Digital ground
Pin 21	Real time clock in	22	Digital ground
Pin 23	—	24	A/D reference
Pin 25	—	26	A/D reference

Recommended range for single ended inputs:
Level > 1 V
Input cable length < 4.5m (15 ft.)

Recommended range for differential inputs:
10 mV < level < 10 V
Cable length as needed, twisted pair with shield

Specifications
Double module (A8000)
2.0 A @ +5 Vdc
1.3 ac bus loads
1.0 dc bus loads
Distribution panel insert: not available

Related Documentation

LSI-11 Analog System User's Guide (EK-AXV11-UG)

AXV11-C ANALOG INPUT/OUTPUT MODULE

Function
Sixteen single-ended or eight differential analog input signals are converted to digital data and interfaced to the LSI-11 bus. Two analog output signals are generated from digital data provided through the LSI-11 bus.

Analog signals can be unipolar (0–10 V) or bipolar (±10 V), and gain of 1, 2, 4, or 8 can be selected under program control for input signals. Resolution in digital format is 12 bits in binary, binary offset, or two's complement form .

Analog input conversions can be started by a program command, an external trigger, or by a real-time clock. I/O is under program control. An interrupt is generated when when A/D conversion is complete. D/A conversion for analog output is started when a value is loaded into the D/A conversion buffer. See Figures C-10 and C-11.

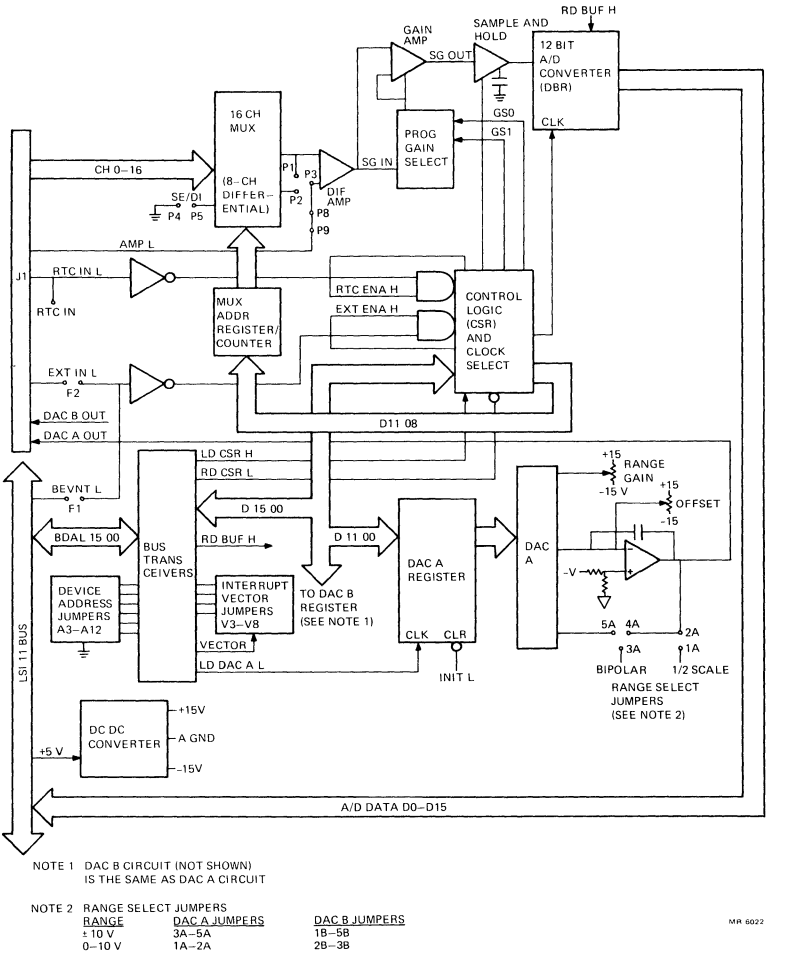


Figure C-10 AXV11 Block Diagram

Programming Interface

The device base address is set with jumpers A12–A3. See Figure C-12.

Control/status register (A/D)	Base address
Data buffer register (A/D)	Base address +2
D/A converter buffer A	Base address +4
D/A converter buffer B	Base address +8

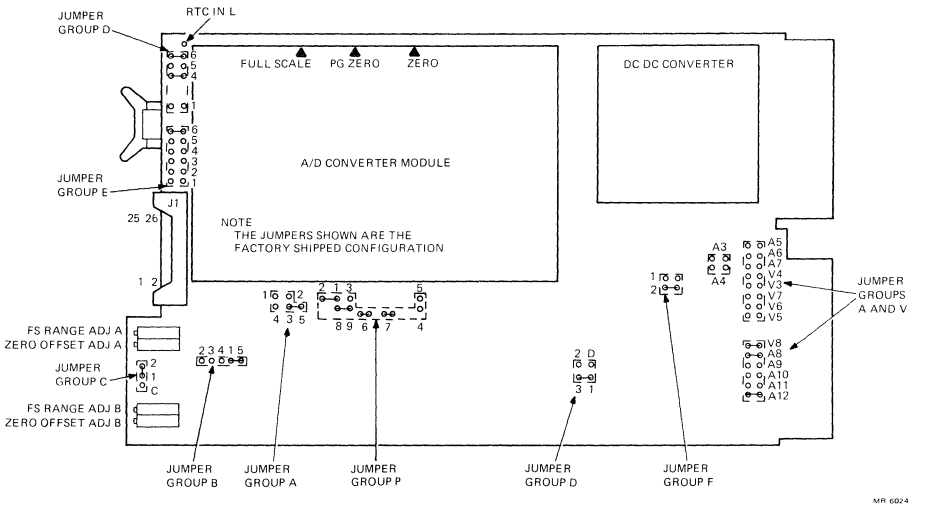


Figure C-11 AXV11 Jumper Locations

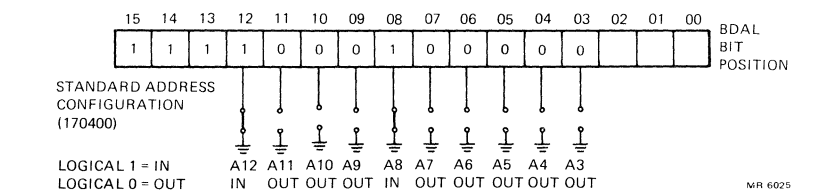


Figure C-12 AXV11 Base Address Selection

The standard base address is 17 770 400_8.

Interrupt vector addresses are set with jumpers V8–V3. See Figure C-13.

A/D done interrupt	Vector address
Error interrupt	Vector address +4

The standard vector address is 400_8.

Physical Interface

AAV11-C connects through a 26-pin jack on the module (3M #3399-7026). No distribution panel insert is currently available for this module.

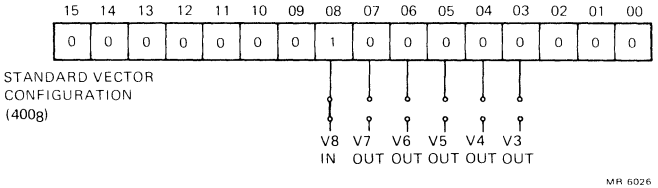


Figure C-13 AXV11 Vector Address Selection

Note: The user is responsible for installation of the AXV11-C module, and for ensuring compliance with any applicable EMI/RFI regulations.

Pin 1	CH0	2	CH8 or Return 0
Pin 3	CH1	4	CH9 or Return 1
Pin 5	CH2	6	CH10 or Return 2
Pin 7	CH3	8	CH11 or Return 3
Pin 9	CH4	10	CH12 or Return 4
Pin 11	CH5	12	CH13 or Return 5
Pin 13	CH6	14	CH14 or Return 6
Pin 15	CH7	16	CH15 or Return 7
Pin 17	Analog ground	18	AMP
Pin 19	External trigger in	20	Digital ground
Pin 21	Real time clock in	22	Digital ground
Pin 23	DAC A return	24	DAC A out
Pin 25	DAC B return	26	DAC B out

Recommended limits for single-ended inputs are:
Level > 1 V
Input cable length <4.5 m (15 ft.)

Recommended limits for differential inputs are:
10 mV < level < 10 V
Cable length: as needed, twisted-pair shielded lines

Specifications
Double module (A0026)
2.0 A @ +5 Vdc
1.3 ac bus loads
1.0 dc bus loads
Distribution panel insert: not available

Related Documentation

LSI-11 Analog System User's Guide (EK-AXV11-UG)

DLV11-E **ASYNCHRONOUS**
DLV11-EP (MICRO/PDP-11) **SERIAL-LINE INTERFACE**

Function

Interfaces asynchronous serial line to the LSI-11 bus. I/O is under program control. Receiver interrupt is generated on change in modem status or when a character has been received. Transmitter interrupt is generated when DLV11-E is ready to send a character. Data rates are program—or jumper—selectable from 50—19.2K baud. Bell type 103, 113, 202C, 202D, and 212 modem control is supported.See Figures C-14, C-15, C-16.

Programming Interface

The device base address is set with jumpers A3–A12. See Figure C-17.

RCSR (R/W)	Base address	Receive Control/Status Register
RBUF (R)	Base address +2	Receive Buffer
XCSR (R/W)	Base address +4	Transmit Control/Status Register
XBUF (W)	Base address +6	Transmit Buffer

The standard base address is 17 775 610_8.

Interrupt vectors are set with jumpers V3–V8. See Figure C-18.
Receive: Vector address
Transmit: Vector address +4
The standard vector address is 300_8.

Physical Interface

DLV11-EP uses a 25-pin RS-232 connector at the distribution panel.

For modem connection to the DLV11-EP distribution panel insert, use BC22E cable. For local null modem connection, use BC22C cable.

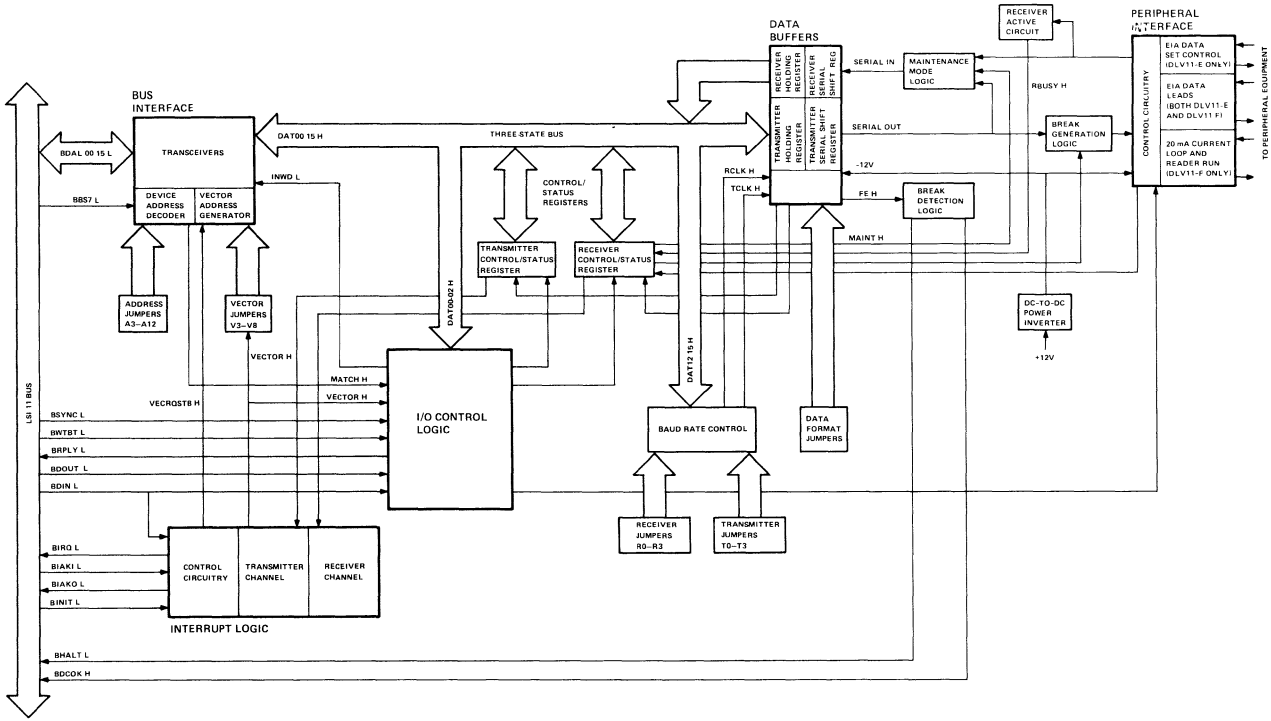


Figure C-14 DLV11-E Block Diagram

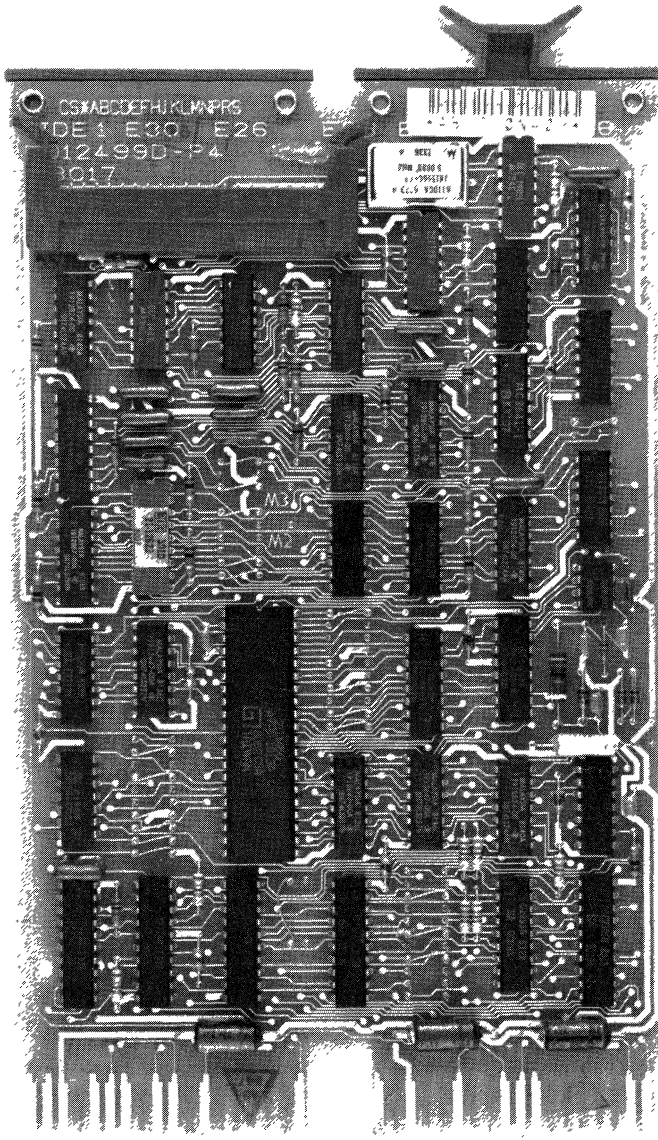


Figure C-15 DLV11-E

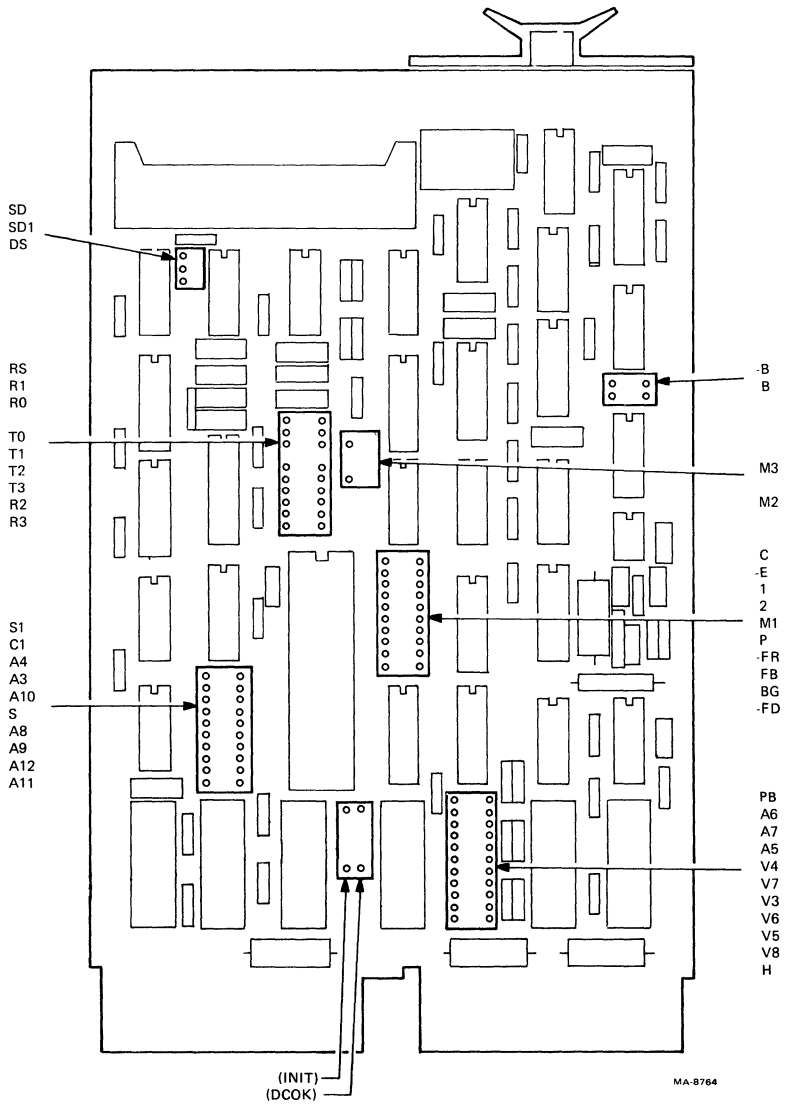


Figure C-16 DLV11-E Jumper Locations

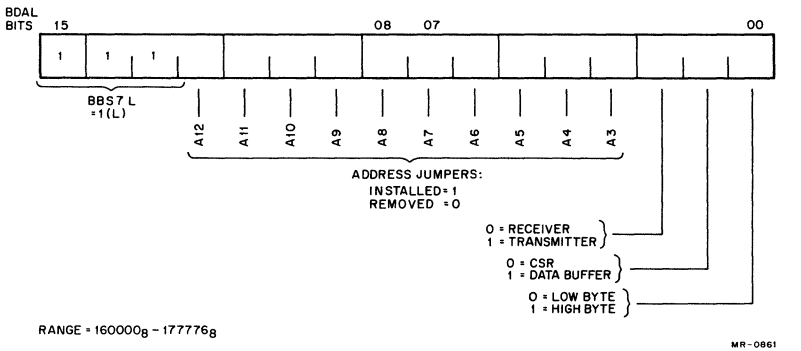


Figure C-17 DLV11-E Base Address Selection

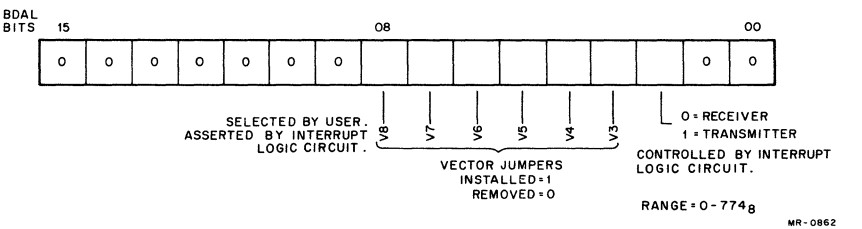


Figure C-18 DLV11-E Vector Selection

Specifications

Double module (M8017)
1.0 A @ +5 Vdc
0.15 A @ +12 Vdc
1.6 ac bus load
1 dc bus load
1 × 4 distribution panel insert

Related Documentation

DLV11-E and DLV11-F Asynchronous Line Interface Users Manual (EK-DLV11-OP), *Field Maintenance Print Set* (MP-00460)

DLV11-J
DLV11-JP (MICRO/PDP-11)

FOUR ASYNCHRONOUS SERIAL LINES

Function

Interfaces four asynchronous serial lines to the LSI-11 bus. Each line has its own set of control/status and data registers. I/O is under program control. Transmit interrupts are generated when a line is ready to transmit, and receive interrupts are generated each time a character is received. See Figures C-19, C-20, and C-21.

Data leads only are supported. Signals conform to RS-232C and RS-423 in the factor configuration. RS-422 can be selected by altering jumpers.

Transmit and receive speeds for each channel are the same. Speeds for the four channels are individually selected by jumpering wire-wrap pin 0-3 (number corresponds to the channel) to the appropriate baud-rate generator pin:

Wirewrap Pin Label	Baud Rate (Bit/S)
U	150
T	300
V	600
W	1,200
Y	2,400
L	4,800
N	9,600
K	19,200
Z	38,400

Programming Interface

The device base address (BA) is set with jumpers A5-A12. See Figure C-22.

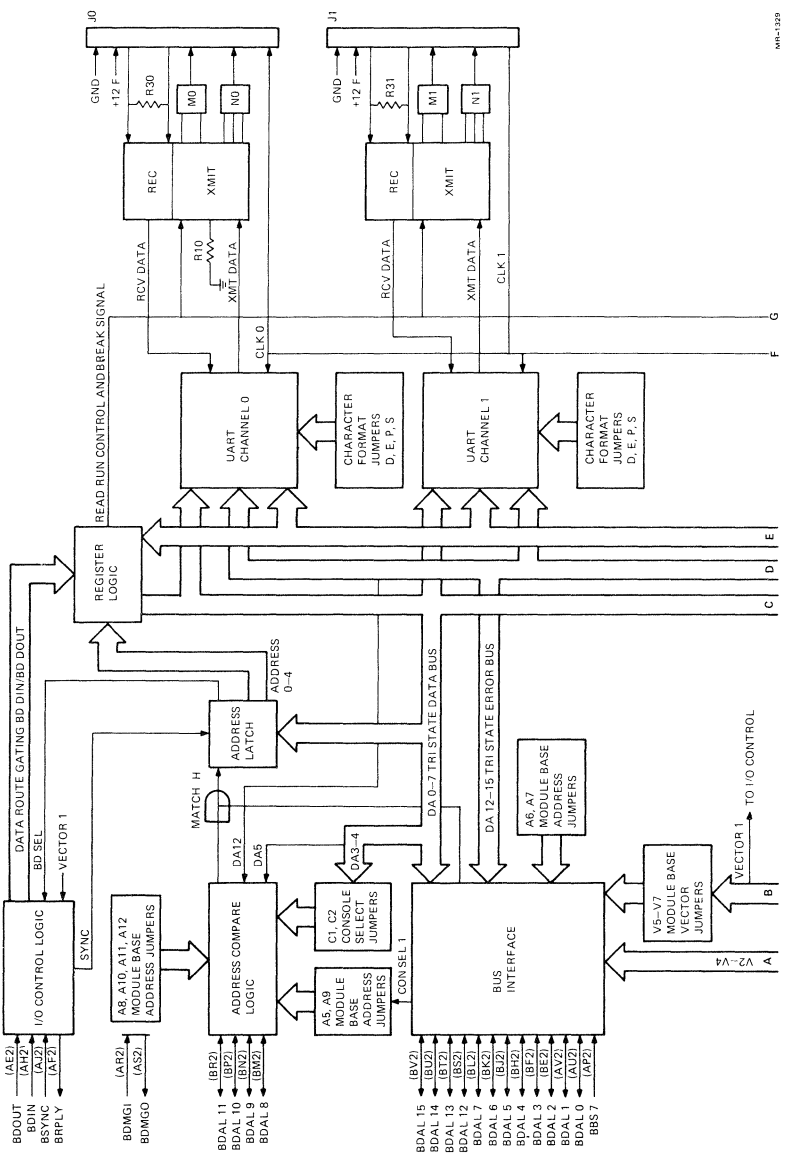
	Channel 0	Channel 1	Channel 2	Channel 3	Channel 3 (Jumpers C1, C2 = 1)
RCSR (R/W)	BA	BA+8	BA+16	BA+24	$17\ 777\ 560_8$
RBUF (R)	BA+2	BA+10	BA+18	BA+26	$17\ 777\ 562_8$
XCSR (R/W)	BA+4	BA+12	BA+20	BA+28	$17\ 777\ 564_8$
XBUF (W)	BA+6	BA+14	BA+22	BA+30	$17\ 777\ 566_8$

The standard base address is $17\ 776\ 500_8$.

The device base interrupt vector (BV) is set with jumpers V3–V8: See Figure C-23.

	Channel 0	Channel 1	Channel 2	Channel 3	Channel 3 (Jumpers C1, C2 = 1)
Receiver	BV	BV+8	BV+16	BV+24	60_8
Transmitter	BV+4	BV+12	BV+20	BV+28	64_8

The standard base vector is 300_8.


J0
J1
GND
+12 F
R30
R31
M0
N0
M1
N1
REC
XMIT
RCV DATA
XMT DATA
R10
CLK 0
CLK 1
READ RUN CONTROL AND BREAK SIGNAL
UART CHANNEL 0
UART CHANNEL 1
CHARACTER FORMAT JUMPERS D, E, P, S
REGISTER LOGIC
DATA ROUTE GATING BD DIN/BD DOUT
BD SEL
VECTOR 1
SYNC
I/O CONTROL LOGIC
ADDRESS LATCH
ADDRESS 0–4
MATCH H
DA12
DA5
DA3–4
A8, A10, A11, A12 MODULE BASE ADDRESS JUMPERS
ADDRESS COMPARE LOGIC
C1, C2 CONSOLE SELECT JUMPERS
A5, A9 MODULE BASE ADDRESS JUMPERS
CON SEL 1
DA 0–7 TRI STATE DATA BUS
DA 12–15 TRI STATE ERROR BUS
A6, A7 MODULE BASE ADDRESS JUMPERS
BUS INTERFACE
V5–V7 MODULE BASE VECTOR JUMPERS
V2–V4
VECTOR 1
TO I/O CONTROL
BDOUT (AE2)
BDIN (AH2)
BSYNC (AJ2)
BRPLY (AF2)
BDMGI (AR2)
BDMGO (AS2)
BDAL 11 (BR2)
BDAL 10 (BP2)
BDAL 9 (BN2)
BDAL 8 (BM2)
BDAL 15 (BV2)
BDAL 14 (BU2)
BDAL 13 (BT2)
BDAL 12 (BS2)
BDAL 7 (BL2)
BDAL 6 (BK2)
BDAL 5 (BJ2)
BDAL 4 (BH2)
BDAL 3 (BF2)
BDAL 2 (BE2)
BDAL 1 (AV2)
BDAL 0 (AU2)
BBS 7 (AP2)
MR-1329

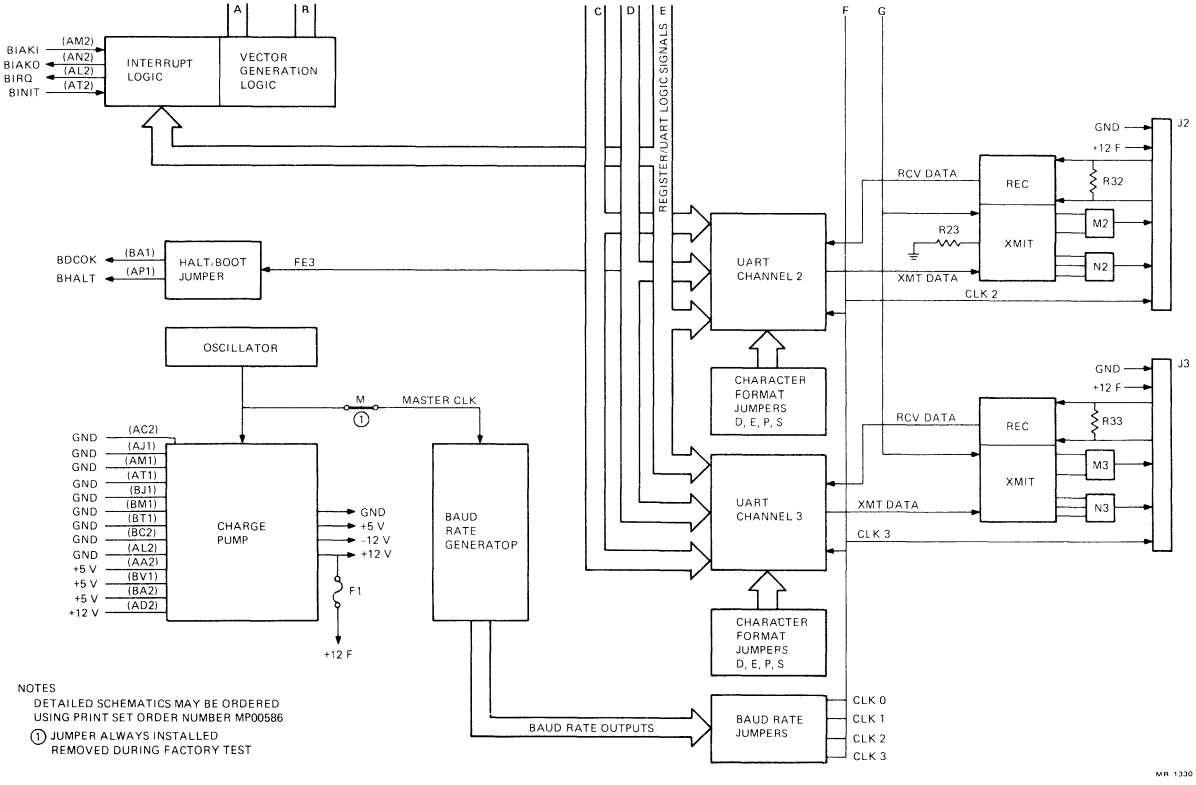


Figure C-19 DLV11-J Block Diagram

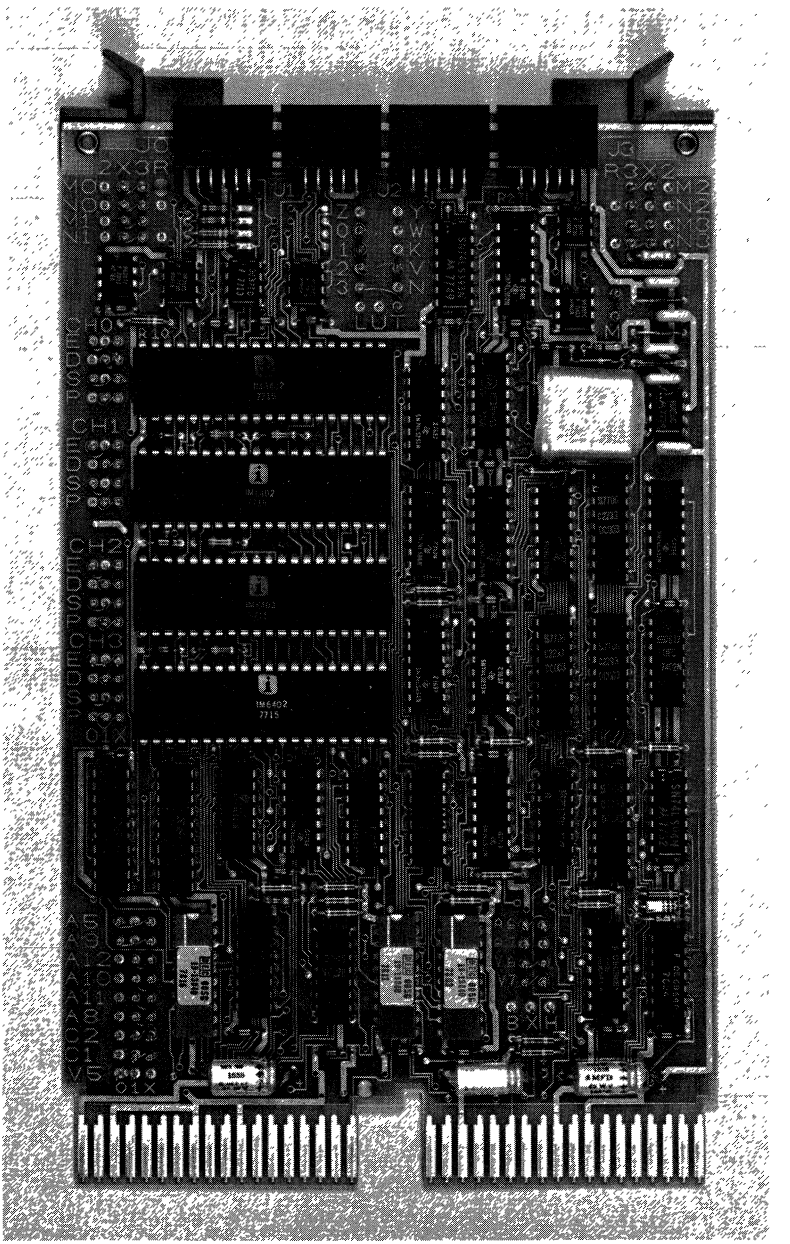


Figure C-20 DLV11-J

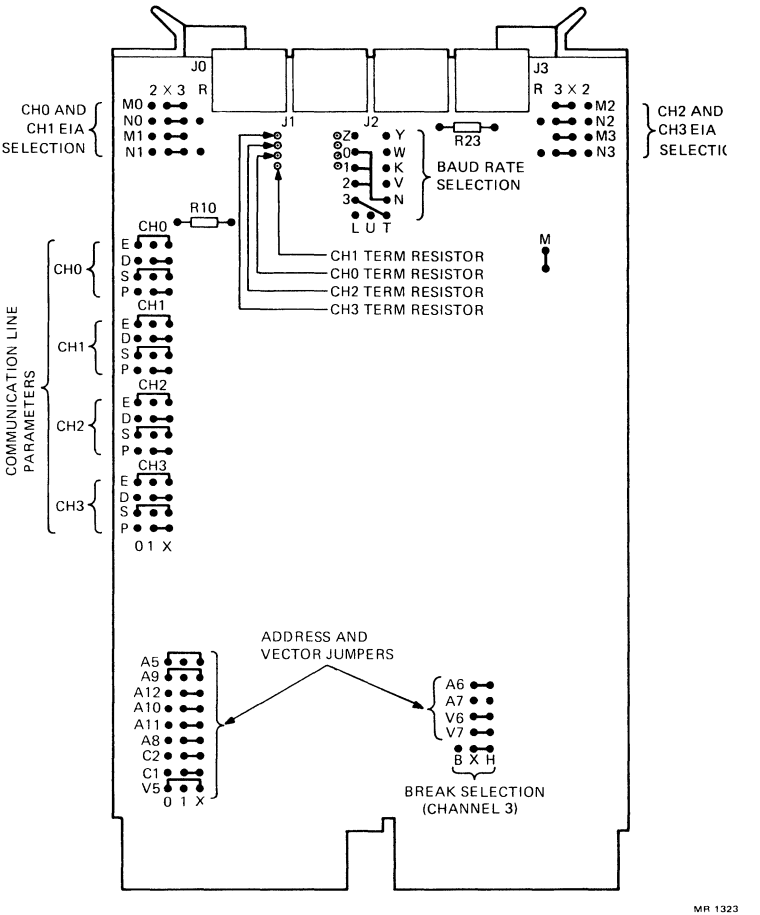


Figure C-21 DLV11-J Jumper Locations

Note: Installation of jumpers C1, C2 enables channel 3 as the console device regardless of jumpers A5–A12 and V3–V8.

Physical Interface

The module connectors are four 10-pin connector blocks. DLV11-JP uses four 25-pin RS-232C connectors at the distribution panel. For modem connection to DLV11-JP, use BC22E cable. For local null modem connection, use BC22D cable.

The factory configuration supports EIA RS-232C and RS-423.

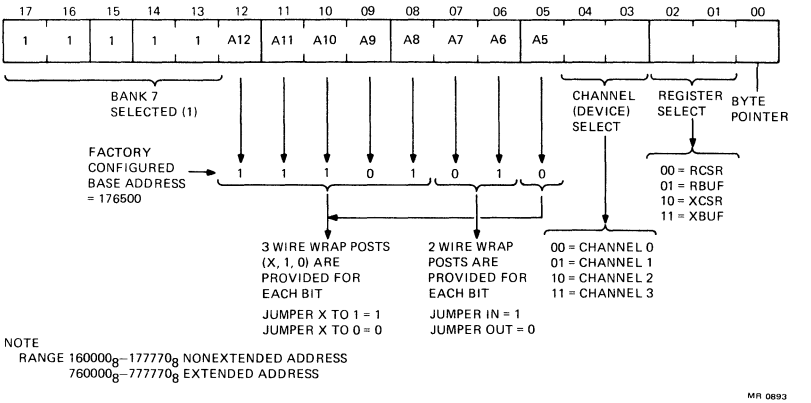


Figure C-22 DLV11-J Base Address Selection

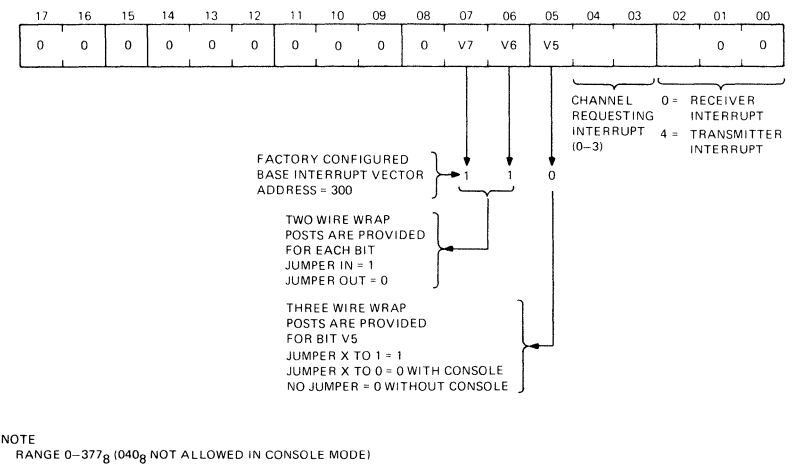


Figure C-23 DLV11-J Vector Selection

Specification

Double module (M8043)
1.0 A @ +5 Vdc
0.25 A @ +12 Vdc
1 ac bus load
1 dc bus load
2 × 3 distribution panel insert

Related Documentation

DLV11-J User's Guide (EK-DLV1J-UG)

Field Maintenance Print Set (MP-00586)

DMV11-AA (-AP, MICRO/PDP-11)
DMV11-AB (-BP, MICRO/PDP-11)
DMV11-AC (-CP, MICRO/PDP-11)
DMV11-FP (MICRO/PDP-11)

DECnet SYNCHRONOUS SERIAL LINE INTERFACE

Function

Transmits and receives data over a synchronous serial line, using the DDCMP (DECnet) protocol. Supports full- or half-duplex communication over point-to-point or multipoint links (other devices on multipoint links must be DMP11's or DMV11's). DMA I/O is used. Message data is set up in a data buffer in memory for transmit operations and read from a data buffer in memory for receive operations. DMV11 handles all aspects of DDCMP processing. See Figures C-24, C-25, and C-26.

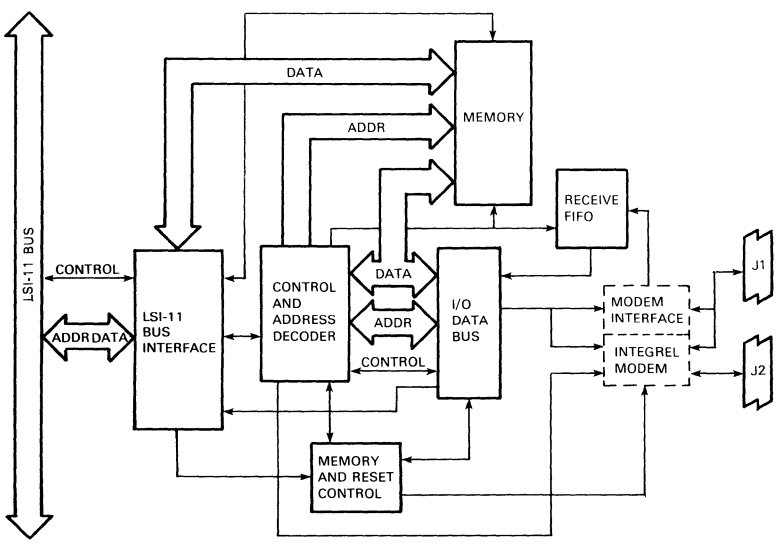


Figure C-24 DMV11 Block Diagram

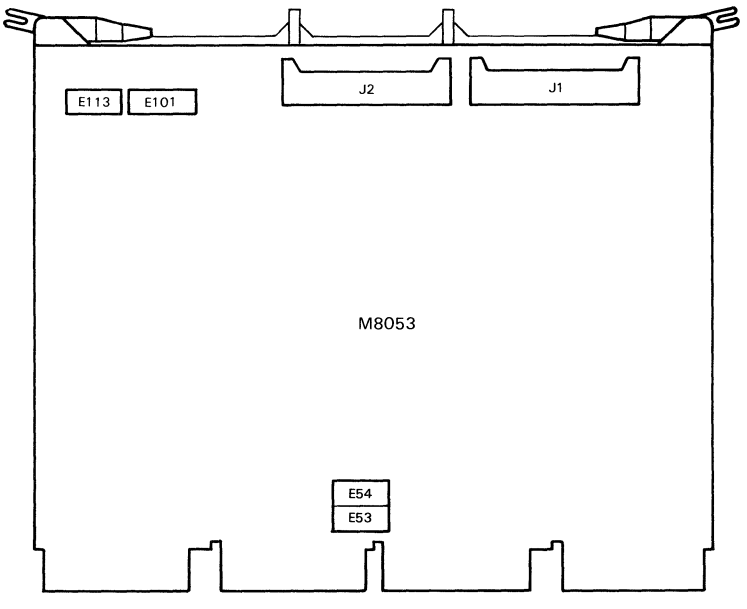


Figure C-25 DMV11-AP, -BP, -FP Switch Locations

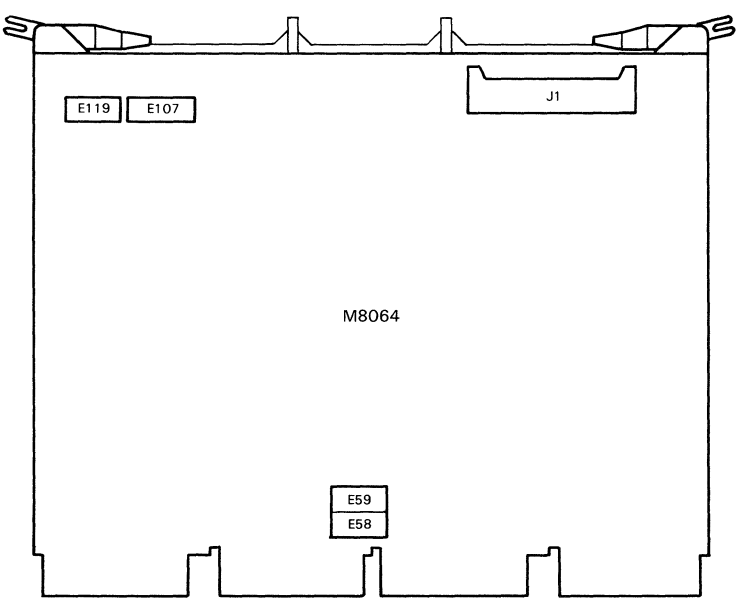


Figure C-26 DMV11-CP Switch Locations

DMV11-AP: EIA RS-232C, CCITT V.24 or V.28 interface standard. Operates to 19.2K bps.

DMV11-BP: CCITT V.35/DDS interface standard. Operates to 56K bps.

DMV11-CP: Includes integral modem for local interconnection. Operates at 56K bps.

DMV11-FP: EIA RS-423A interface standard. Operates to 56K bps.

Programming Interface

Device addresses are set using switches E53 <8:1> and E54 <2:1> (for DMV11-AP, -BP, -FP), or switches E58 <8:1> and E59 <2:1> (for DMV11-CP). See Figure C-27.

The vector address is set using switches E54 <8:3> (for DMV11-AP, -BP, -FP) or E59 <8:3> (for DMV11-CP). See C-28.

Device and vector addresses are assigned according to the floating convention (see Appendix D).

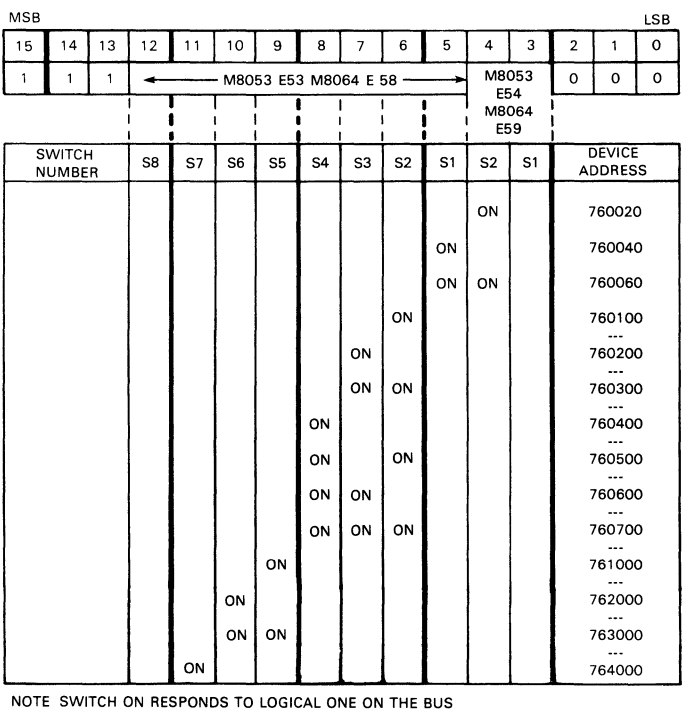


Figure C-27 DMV11 Base Address Selection

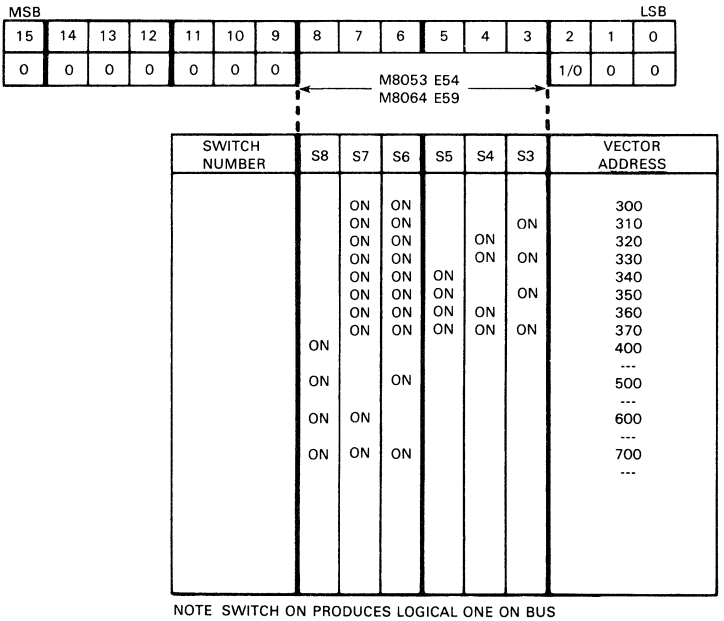


Figure C-28 DMV11 Vector Selection

Physical Interface

DMV11-AP uses a single 25-pin RS-232 connector at the distribution panel. Use BC22B cable.

DMV11-BP uses a single V.35/DDS connector at the distribution panel. The external cable is not available from DIGITAL.

DMV11-CP uses four connectors at the distribution panel. One BC55M cable is used for half-duplex operation; two BC55M cables for full-duplex operation. In point-to-point links and for the final node in a multipoint link, H3258 and H3257 terminators (included) are installed on the unused connectors.

DMV11-FP uses a single 37-pin RS-449 connector at the distribution panel. The external cable is not available from DIGITAL.

Specifications:

Quad Module (M8053: -AP, -BP, -FP) (M8054: -CP)
3.4 A @ +5 Vdc (-AP,-BP,-FP)
3.35 A @ +5 Vdc (-CP)
0.38 A @ +12 Vdc (-AP,-BP,-FP)

0.26 A @ +12 Vdc (-CP)
2 ac bus loads
1 dc bus load
1 × 4 distribution panel insert (-AP,-BP,-FP)
2 × 3 distribution panel insert (-CP)

Related Documentation

DMV11 Synchronous Controller User's Guide (EK-DMV11-UG)
DMV11 Synchronous Controller Technical Manual (EK-DMV11-TM)
Field Maintenance Print Set (MP-00942)

DPV11 DPV11-DP (MICRO/PDP-11) — SYNCHRONOUS SERIAL LINE INTERFACE

Function

Transmits and receives data over a synchronous serial line, using bit- or character-oriented protocols. Interfaces to modems using RS-232C or RS-449 standards, with full Category I and partial Category II modem control. Full- or half-duplex operation, 6 to 8 bit characters (in character-oriented protocol mode), data rates up to 56K baud. Cyclic Redundancy Code (CRC) generation and checking is performed by the DPV11 (not usable for BISYNC protocol). See Figures C-29, C-30, C-31.

Programming Interface

Device addresses are set using jumpers W29-W35. See Figure C-32.

RXCSR (R/W)	Base address	Receive control and status
RDSR (R)	Base address +2	Receive data and status
PCSAR (W)	Base address +2	Parameter control, sync/address
PCSCR (R/W)	Base address +4	Parameter control and character length
RDSR (R/W)	Base address +6	Transmit data and status

The standard base address is 17 7600 10_8.

The interrupt vector address is set with jumpers W40–W46. See Figure C-33.

The standard vector assignment is 300_8.

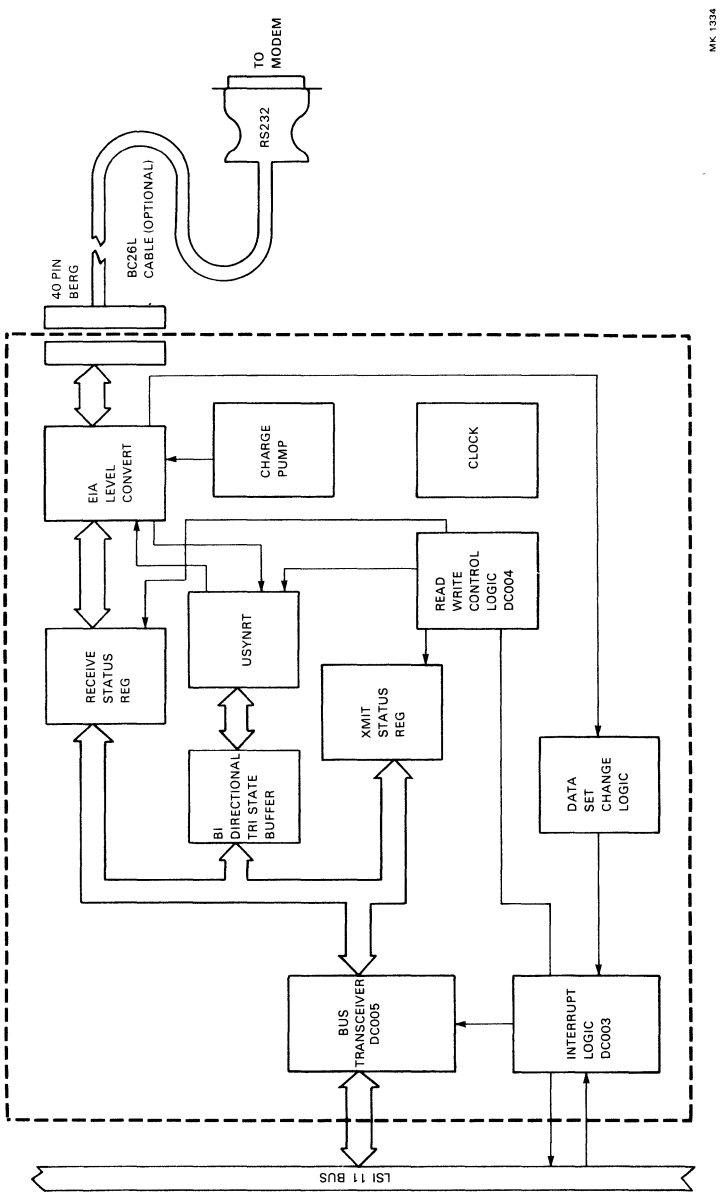


Figure C-29 DPV11 Block Diagram

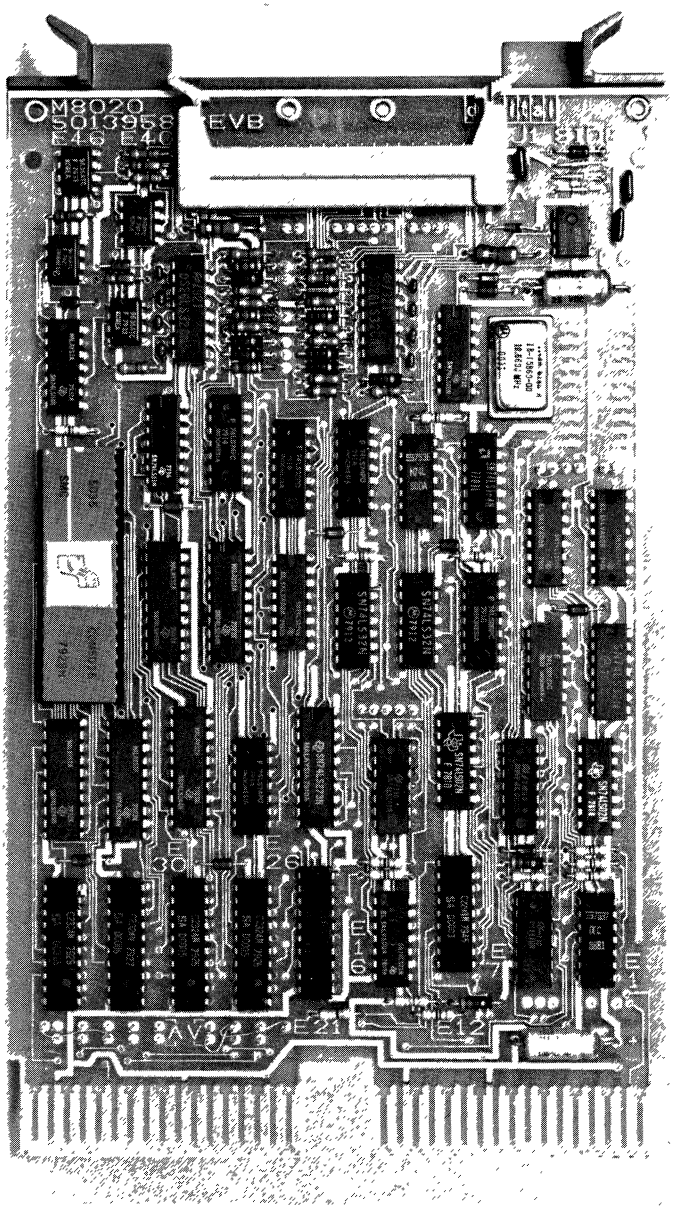


Figure C-30 DPV11

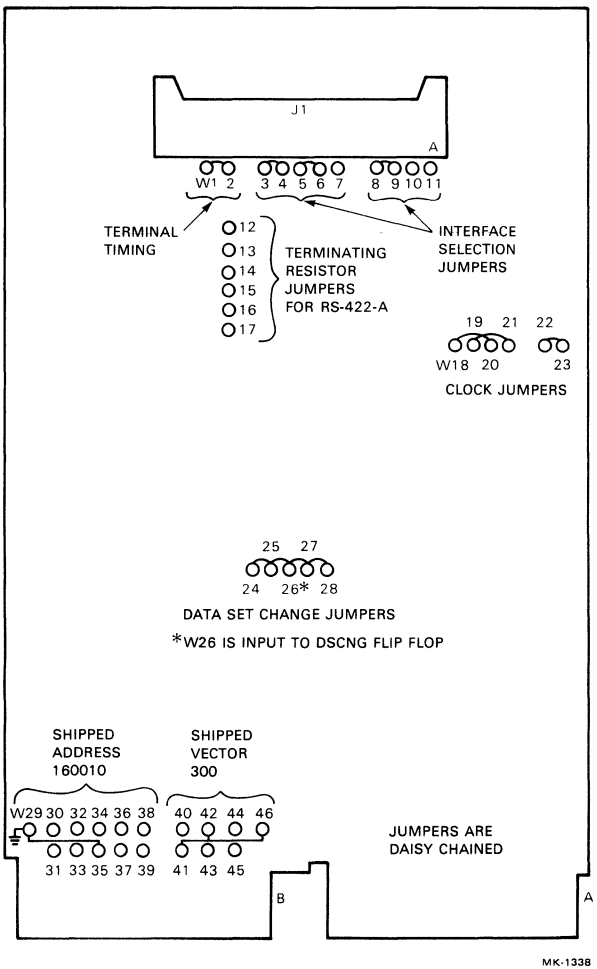


Figure C-31 DPV11 Jumper Locations

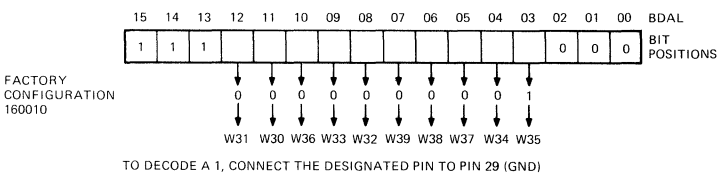


Figure C-32 DPV11 Base Address

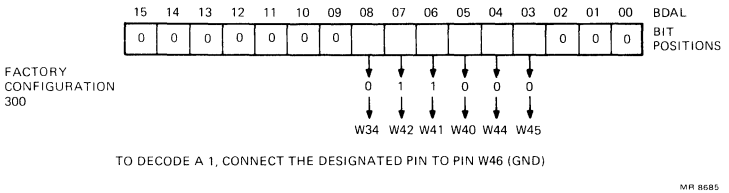


Figure C-33 DPV11 Vector Address

Physical Interface

DPV11-DP has a single 25-pin RS-232 connector at the distribution panel. As shipped, DPV11-DP is electrically compatible with RS232 and RS-423A.

Use BC22F cable.

Specifications

Double module (M8020)
1.2 A @ +5 Vdc
0.3 A @ +12 Vdc
1 ac bus load
1 dc bus load
1 × 4 distribution panel insert

Related Documentation

DPV11 User Manual (EK-DPV11-UG)
DPV11 Technical Manual (EK-DPV11-UG)
Field Maintenance Print Set (MP-00919)

DRV11
DRV11-LP (FOR MICRO/PDP-11)

GENERAL-PURPOSE PARALLEL LINE INTERFACE

Function

General-purpose interface to LSI-11 bus. Sixteen input lines and 16 output lines, corresponding to the 16 data lines of the bus, plus control signals for handshaking with user device, all at TTL level.

Two lines from the user device (REQ A and REQ B) are capable of generating processor interrupts. See Figures C-34, C-35, and C-36.

Programming Interface

The device addresses are set with jumpers, A12-A3. See Figure C-37.

DRCSR (R/W)	Base address	Control/Status Register
DROUTBUF (R/W)	Base address +2	Output Buffer
DRINBUF (R)	Base address +4	Input Buffer

The standard base addresses are 17 767 770_8 for the first DRV11 and 17 767 760_8 for the second.

The interrupt vectors are set through jumpers V7–V3. See Figure C-38.

Interrupt A	Vector address
Interrupt B	Vector address +4

The standard vector addresses are 300_8 for the first DRV11 and 310_8 for the second.

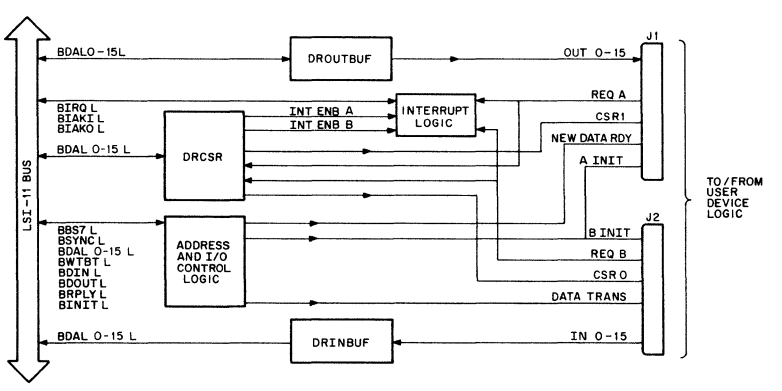


Figure C-34 DRV11 Block Diagram

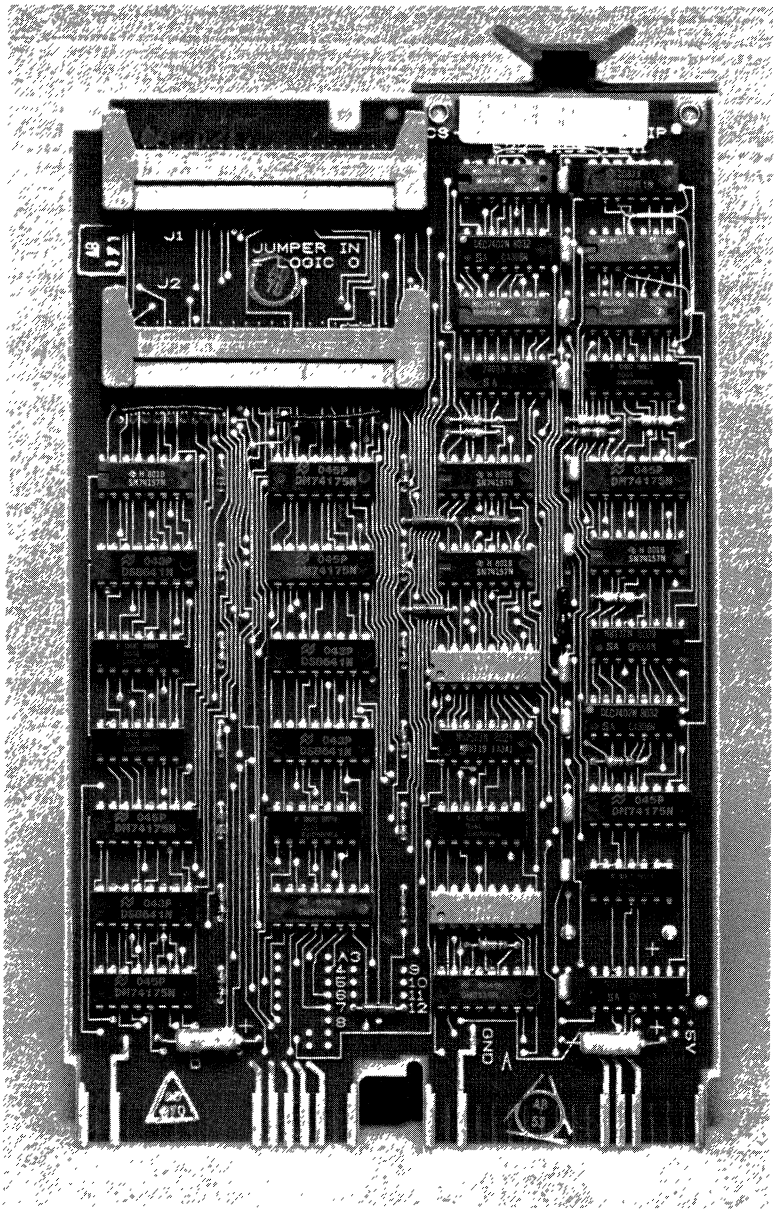


Figure C-35 DRV11

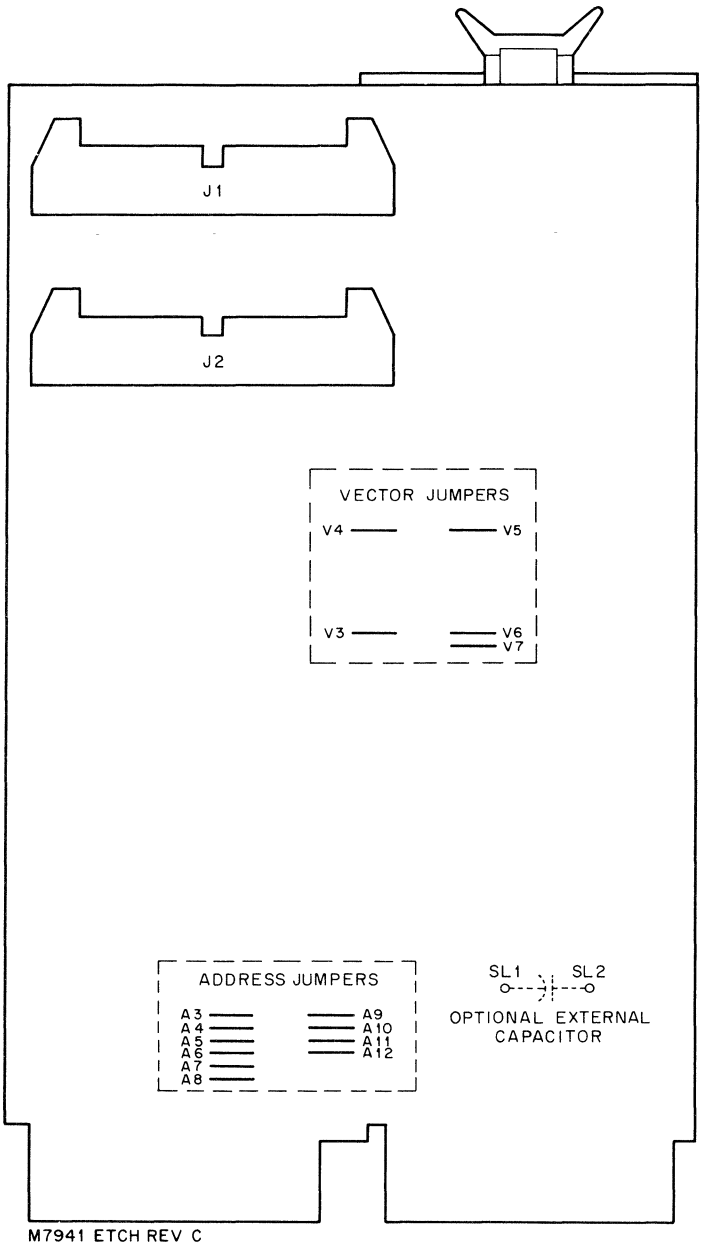


Figure C-36 DRV11 Jumper Location

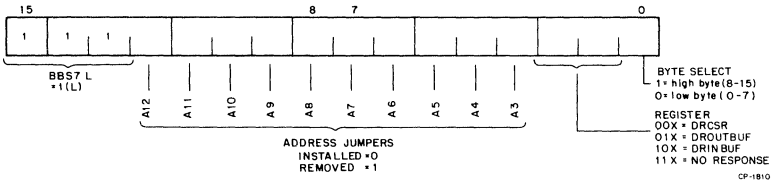


Figure C-37 DRV11 Base Address Selection

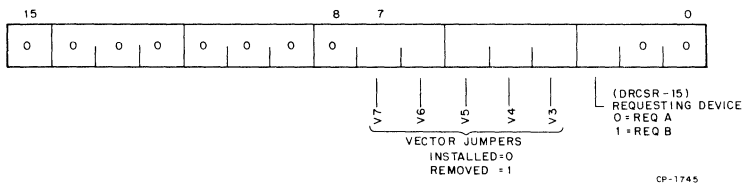


Figure C-38 DRV11 Vector Address Selection

Physical Interface

DRV11-LP uses two 40-pin male connectors at the distribution panel.

Use BC04Z (shielded) or BC07D (unshielded) cables. Both are unterminated at the user device end. See Figure C-39.

Inputs:			Outputs:		
Signal	**Connector**	**Pin**	**Signal**	**Connector**	**Pin**
IN00	J2	TT	OUT00	J1	C
IN01	J2	LL	OUT01	J1	K
IN02	J2	H,E	OUT02	J1	NN
IN03	J2	BB	OUT03	J1	U
IN04	J2	KK	OUT04	J1	L
IN05	J2	HH	OUT05	J1	N
IN06	J2	EE	OUT06	J1	R
IN07	J2	CC	OUT07	J1	T
IN08	J2	Z	OUT08	J1	W
IN09	J2	Y	OUT09	J1	X
IN10	J2	W	OUT10	J1	Z
IN11	J2	V	OUT11	J1	AA
IN12	J2	U	OUT12	J1	BB
IN13	J2	P	OUT13	J1	FF
IN14	J2	N	OUT14	J1	HH
IN15	J2	M	OUT15	J1	JJ
REQ B	J2	S	REQ A	J1	LL
DATA	J2	C	NEW DATA	J1	VV
TRANS	—	—	RDY	—	—
CSRO	J2	K	CSR1	J1	DD
INIT	J2	RR,NN	INIT	J1	P

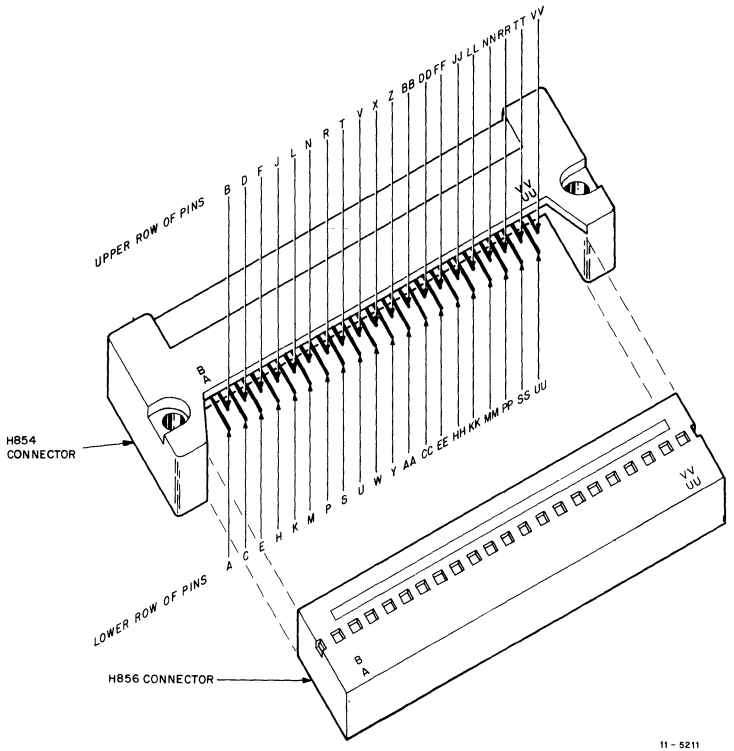


Figure C-39 DRV11 I/O Connector

Specifications

Double module (M7941)
0.9 A @ +5 Vdc
2.8 ac bus loads
1.0 dc bus loads
Two 1 × 4 distribution panel inserts

Related Documentation

ADV11-A, DWV11-A, AAV11-A, DRV11 User's Manual (EK-ADV11-OP)

Field Maintenance Print Set (MP-00054)

DRV11-B **GENERAL-PURPOSE**
DRV11-BP (FOR MICRO/PDP-11) **DMA INTERFACE**

Function

Interfaces 16 input lines and 16 output lines at TTL level, plus control signals, to the LSI-11 bus. DMA I/O: Programmed by processor to move variable length blocks of 8-bit bytes or 16-bit words to or from specified memory locations. Once programmed, no processor intervention is required.

The DRV11-B interface operates as both a slave and a master device. Prior to becoming bus master, all data transfers out (DATO) or data transfers in (DATI) are in respect to the processor. Once the DRV11-B is granted bus mastership by the processor, all data transfers are in respect to the DRV11-B. See Figure C-40.

DMA operation is initialized under program control by loading the WCR with the two's complement of the number of words to be transferred, loading the BAR with the first address to or from which data is to be transferred, or loading the CSR with the desired function bits. After the interface is initialized, data transfers are under control of the DMA logic.

Program Control Transfers— Data transfers may be performed under program control by addressing the IDBR or ODBR and reading or writing data.

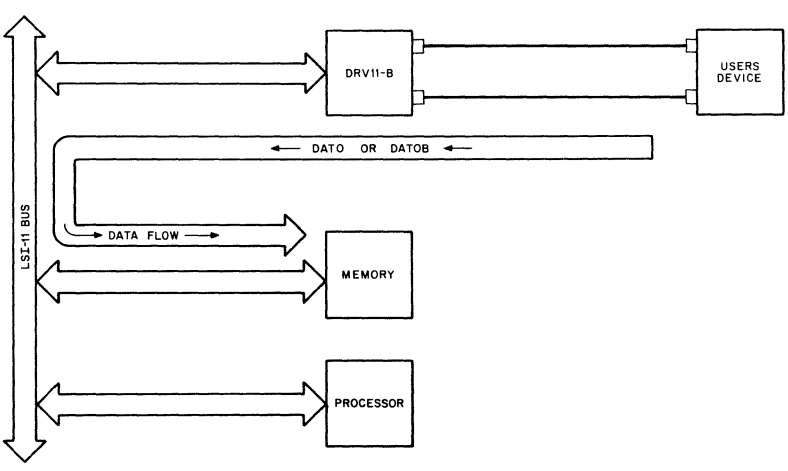


Figure C-40 DRV11-B DMA I/O

DMA Control Transfers— DMA input (DATI) or output (DATO) data transfers occur when the processor clears READY. For a DATO cycle (DRV11-B to memory transfer), the user's I/O device presets the control bits [word count increment enable (WC INC ENB), but address increment enable (BA INC ENB), C1, C0, A00 and ATTN], and asserts CYCLE REQUEST to gain use of the LSI-11 bus. When CYCLE REQUEST is asserted, input data is latched into the input DBR, the control bits are latched into the DRV11-B DMA control and BUS goes low. A DATI cycle—memory to DRV11-B transfer—is handled in a similar manner, except that the output data is latched into the output DBR at the end of the bus cycle.

When the DRV11-B becomes bus master, a DATO or DATI cycle is performed directly to or from the memory location specified by the BAR. At the end of each cycle, the WCR and BAR are incremented and BUSY goes high while READY remains low. A second DATO or DATI cycle is performed when the user's I/O device again asserts CYCLE REQUEST. DMA transfers will continue until the WCR increments to zero, at which time READY goes high and the DRV11-B generates an interrupt (if interrupt enable is set) to the processor.

If burst mode is selected (SINGLE CYCLE low), only one CYCLE REQUEST is required for the complete transfer of the specified number of data words. See Figures C-41, C-42, and C-43.

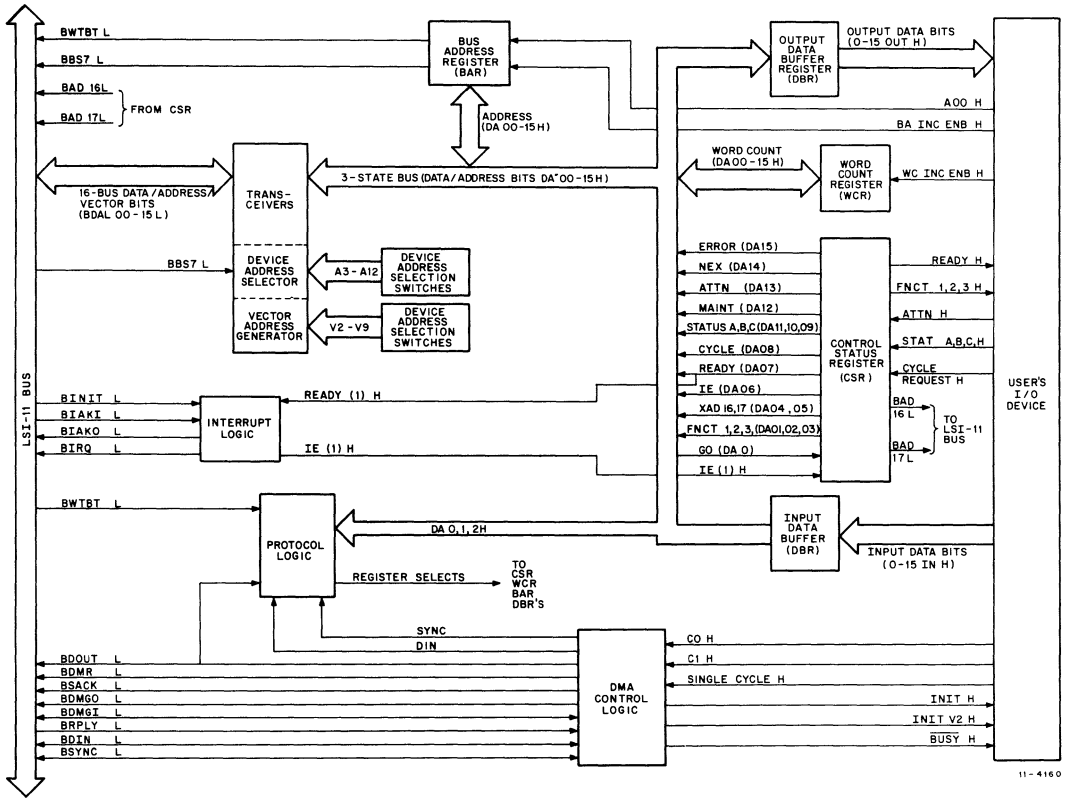


Figure C-41 DRV11-B Block Diagram

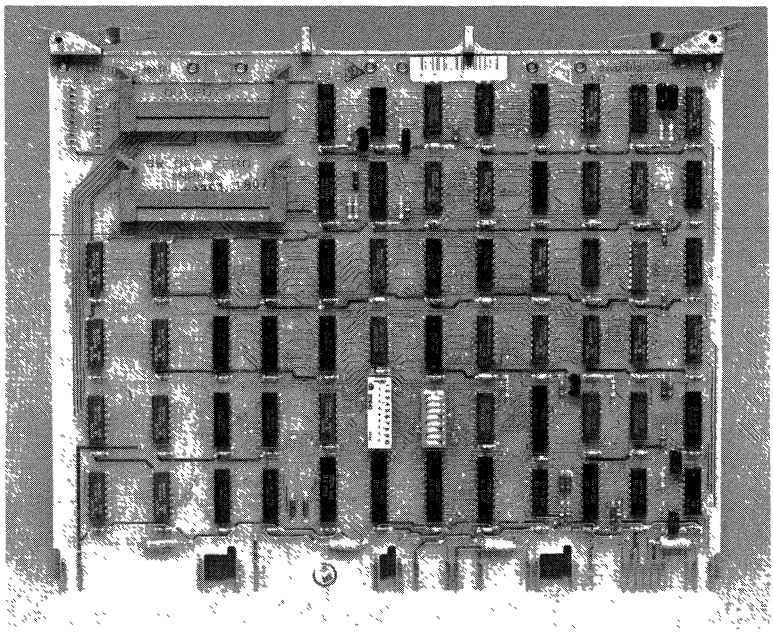


Figure C-42 DRV11-B

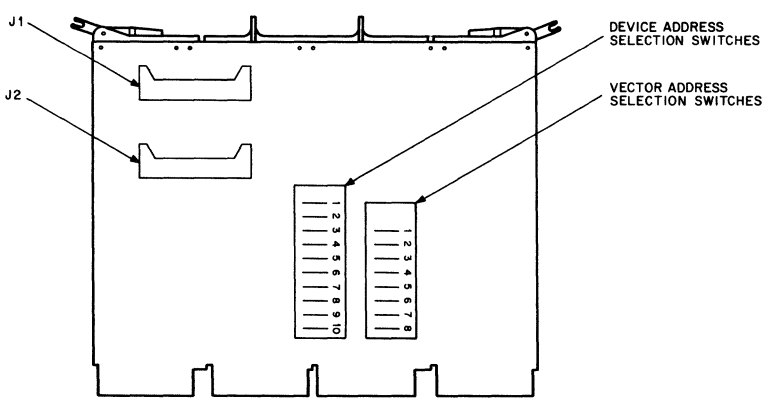


Figure C-43 DRV11-B Switch Locations

Programming Interface

The device base address is set through ten switches in switch pack S2. See Figure C-44.

Standard Addresses:

WCR (R/W)	Base address	Word Count Register
BAR (R/W)	Base address +2	Bus Address Register
CSR (R/W)	Base address +4	Control/Status Register
IDBR (R)	Base address +6	Input Data Buffer
ODBR (W)	Base address +6	Output Data Buffer

The standard base address is 17 772 410_8.

The interrupt vector is set through eight switches in switch pack S1. See Figure C-45.

The standard vector is 124_8.

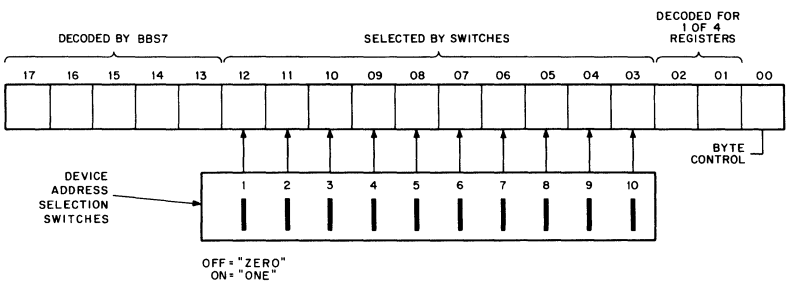


Figure C-44 DRV11-B Base Address Selection

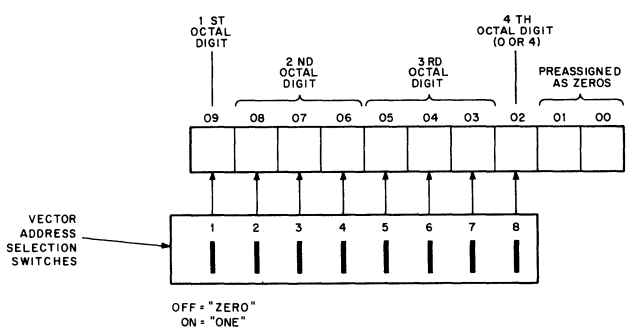


Figure C-45 DRV11-B Vector Address Selection

Physical Interface

J1 (output) and J2 (input) are 40-pin male connectors. DRV11-BP repeats these connectors at the distribution panel. See Figure C-46. Suitable cables include BC04Z and BC08R, and may be up to 15.2 m (50 feet) long.

Connector Signals:	Output (J1)	Input (J2)
B	CYCLE REQUEST	BUSY
D	INIT V2	ATTN
F	READY	AOO
J	WC INC ENB	BA INC ENB
K	SINGLE CYCLE	FNCT 3
L	STATUS A	FNCT 3
N	INIT	CO
R	STATUS B	FNCT 2
T	STATUS C	C1
V	STATUS C	FNCT 1
DD	08 OUT	08 IN
FF	09 OUT	09 IN
JJ	10 OUT	10 IN
LL	11 OUT	11 IN
NN	12 OUT	12 IN
RR	13 OUT	13 IN
TT	14 OUT	14 IN
VV	15 OUT	15 IN
CC	07 OUT	07 IN
EE	06 OUT	06 IN
HH	05 OUT	05 IN
KK	04 OUT	04 IN
MM	03 OUT	03 IN
PP	02 OUT	02 IN
SS	01 OUT	01 IN
VV	00 OUT	00 IN

00 OUT - 15 OUT	Data output lines
00 IN - 15 IN	Data input lines
Status A, B, C	User-defined lines from user device
FNCT 1, 2, 3	User-defined lines to user device
INIT	Initializes User Device
INIT V2	Asserted when INIT is asserted or when FNCT2 is written to a one

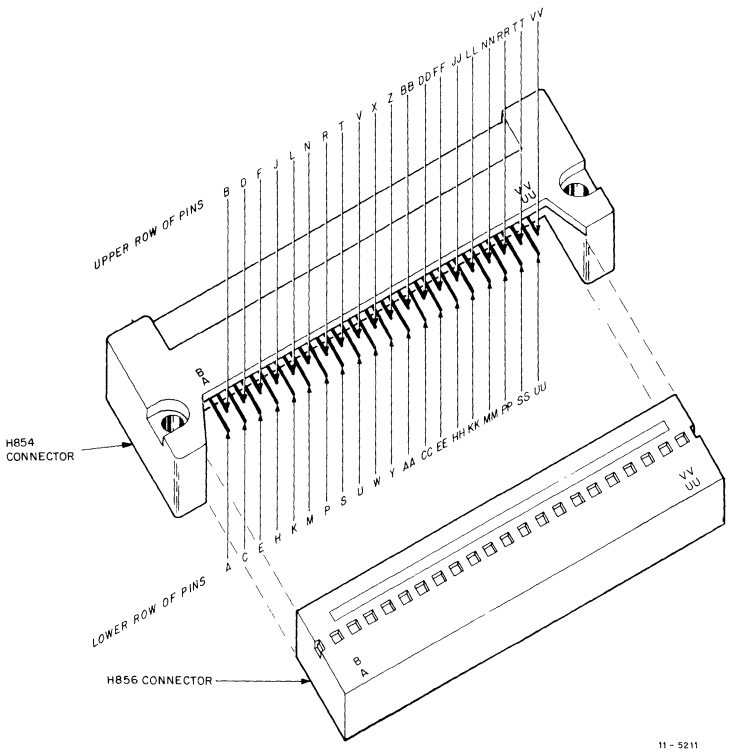


Figure C-46 DRV11-B Connectors

A00	High for word transfers control, address bit <0> during byte transfers
BUSY	Low when DRV11-B is requesting bus control or DMC cycle in progress
READY	Enables DMA transfers from user device when low
C0, C1	User device control of bus cycle

C0	C1	
0	0	DATI
0	1	DATIO
1	0	DATO
1	1	DATOB

Single Cycle	High for normal DMA, low for bus+ mode
WC INC ENB	Inhibits WCR incrementing when low
BA INC ENB	Inhibits BAR incrementing when low
Cycle Request	Low-to-high transition initiates DMA request
ATTN	Driven high to terminate DMA transfers, to set READY, to request interrupt

Specifications

Quad module (M7950)
1.9 A @ +5 Vdc
3.3 ac bus loads
1.0 dc bus loads
Two 1 × 4 distribution panel inserts

Related Documentation

DRV11-B General Purpose DMA Interface User's Manual (EK-DRV1B-OP-001)

Field Maintenance Print Set (MP-00160)

DRV11-J
DRV11-JP (FOR MICRO/PDP-11)

GENERAL-PURPOSE PARALLEL LINE INTERFACE

Function

Interfaces 64 input/output lines to the LSI-11 bus. Line direction is selectable under program control in groups of 16. I/O is under program control. Interrupts are generated by up to 16 of the user device controlled signals. See Figures C-47, C-48, and C-49.

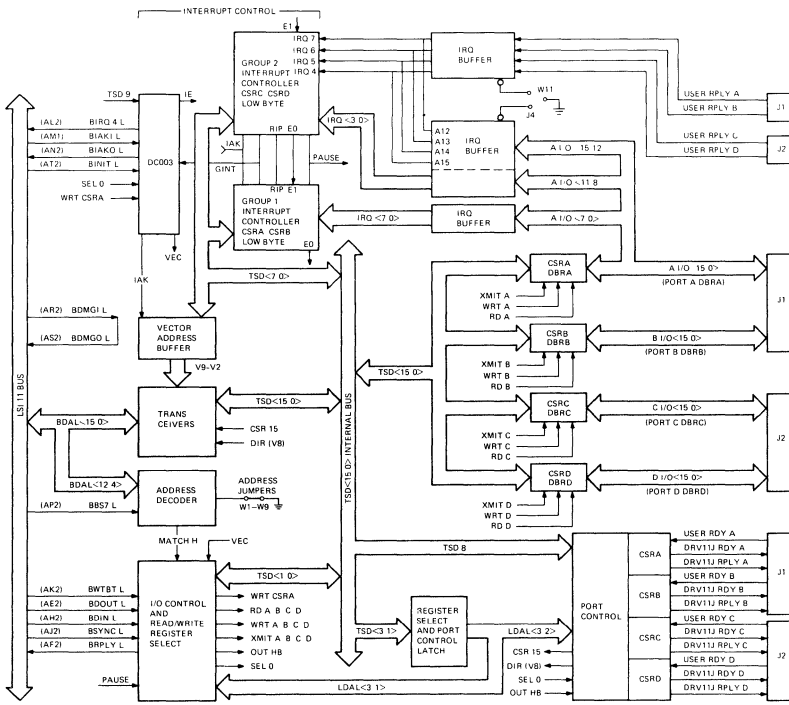


Figure C-47 DRV11-J Block Diagram

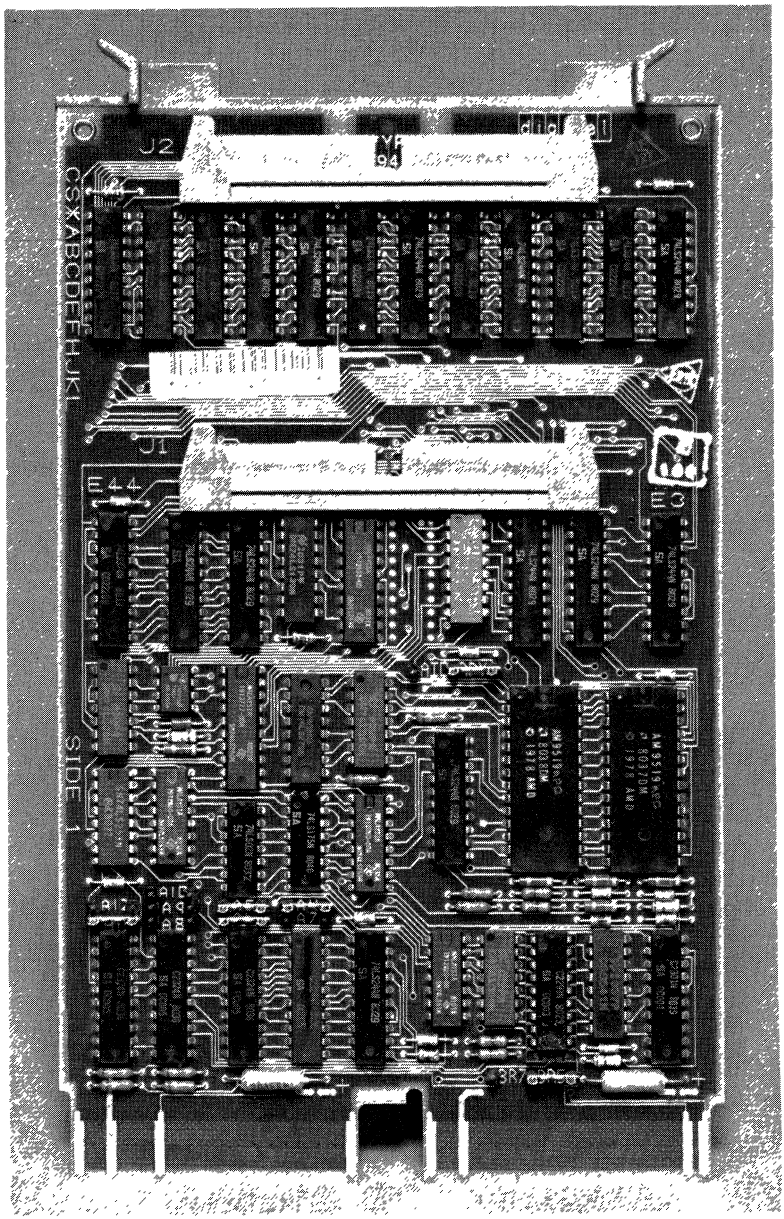


Figure C-48 DRV11-J

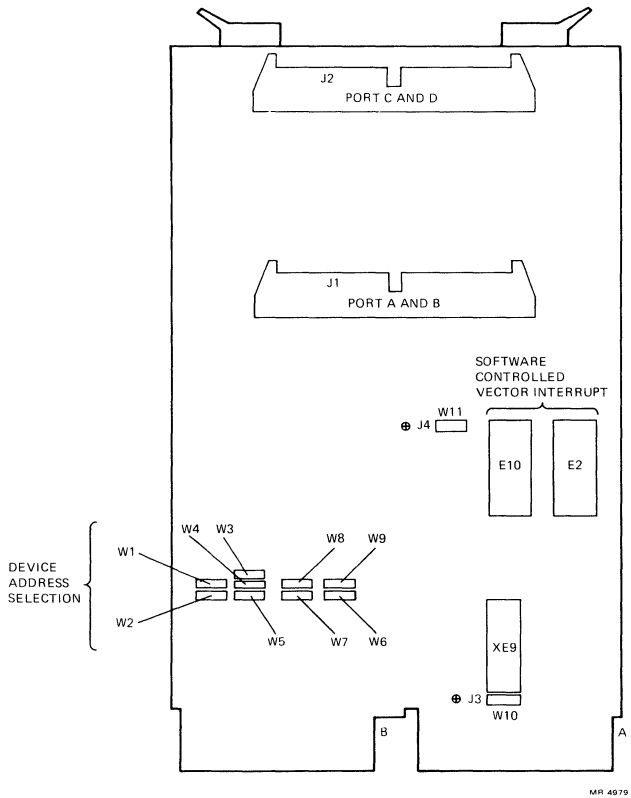


Figure C-49 DRV11-J Jumper Locations

Programming Interface

The device addresses are set with jumpers W1–W9. See Figure C-50.

Standard Addresses:

CSRA (R/W)	Base address	Control/Status Register A
DBRA (R/W)	Base address +2	Data Buffer A
CSRB (R/W)	Base address +4	Control/Status Register B
DBRB (R/W)	Base address +6	Data Buffer B
CSRC (R/W)	Base address +8	Control/Status Register C
DBRC (R/W)	Base address +10	Data Buffer C
CSRD (R/W)	Base address +12	Control/Status Register D
DBRD (R/W)	Base address +14	Data Buffer D

The standard base address is 17 764 160_8.

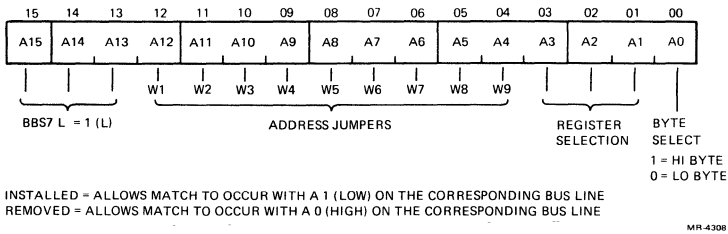


Figure C-50 DRV11-J Base Address Selection

A second module should use 17 764 140_8 and a third 17764120_8.

Interrupt vectors are programmed into a vector address memory through the control/status registers. Standard vector addresses have not been assigned.

Physical Interface

J1 (ports A and B) and J2 (ports C and D) are 50-pin male connectors. DRV11-JP repeats these connectors at the distribution panel. Suitable cables include BC02D and BC08Y.

Specification

Double module (M8049)
1.8 A @ +5 V
2 ac bus loads
1 dc bus load
Two 1 × 4 distribution panel inserts

Related Documentation

DRV11-J User Guide (EK-DRV11J-UG)

Field Maintenance Print Set (MP-00866)

DUV11 SYNCHRONOUS SERIAL
DUV11-DP (FOR MICRO/PDP-11) LINE INTERFACE

Function

Using character-oriented protocols, transmits and receives data over a synchronous serial line. Interfaces to Bell 201 (or equivalent) modem. Full- or half-duplex operation, 5 to 8 bit characters plus parity bit, at up to 9600 baud. Program control I/O: interrupts are generated when synchronized data is receive d, when the transmitter is ready, when the modem status changes, and when a sync or fill character is being sent to the modem.

In internal synchronous mode, DUV11 checks for two contiguous SYNC characters on incoming data. In external synchronous mode, it asserts the Search Sync condition and interrupts the processor when sync is detected. In isochronous mode, each chara cter is preceded by a start bit and followed by a stop bit as in asynchronous communication. See Figures C-51, C-52, and C-53.

Programming Interface

The device base address is set using switches E38<1:8> and E39<1:2>. See Figure C-54.

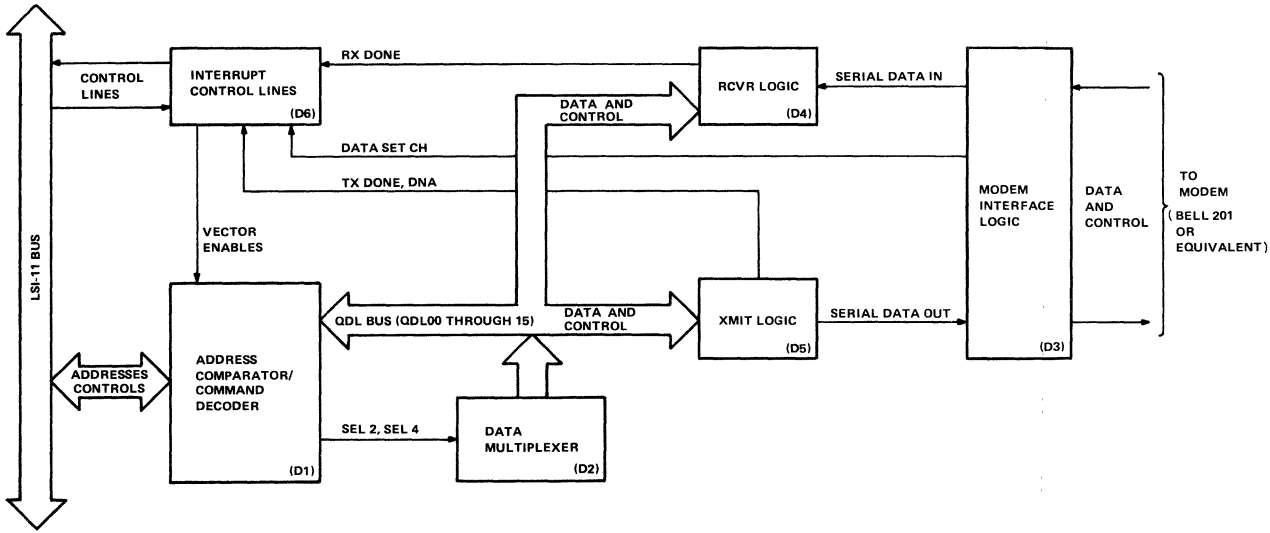
RXCSR (R/W)	Base address	Receiver Status
RXDBUF (R)	Base address +2	Receiver Data Buffer
PARCSR (W)	Base address +2	Parameter Status
TXCSR (R/W)	Base address +4	Transmitter Status
TXDBUF (W)	Base address +6	Transmitter Data Buffer

The interrupt vector is set using switches E39<3:8>. See Figure C-55.

Both the address and vector are assigned using the floating convention (see Appendix D).

Physical Interface

DUV11-DP terminates in a 25-pin RS-232C connector at the distribution panel. Use BC22F cable.



NOTE:
( ) indicates engineering drawing sheet of logical block.

11-5151

Figure C-51 DUV11 Block Diagram

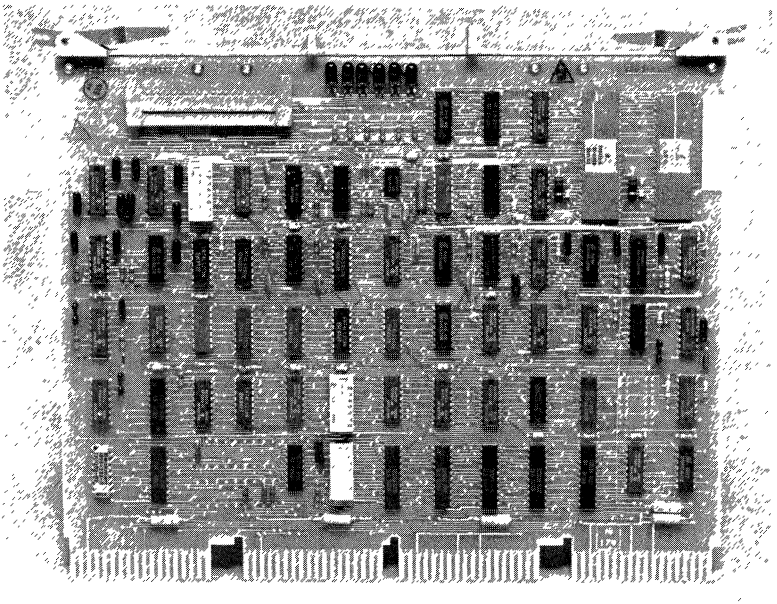


Figure C-52 DUV11

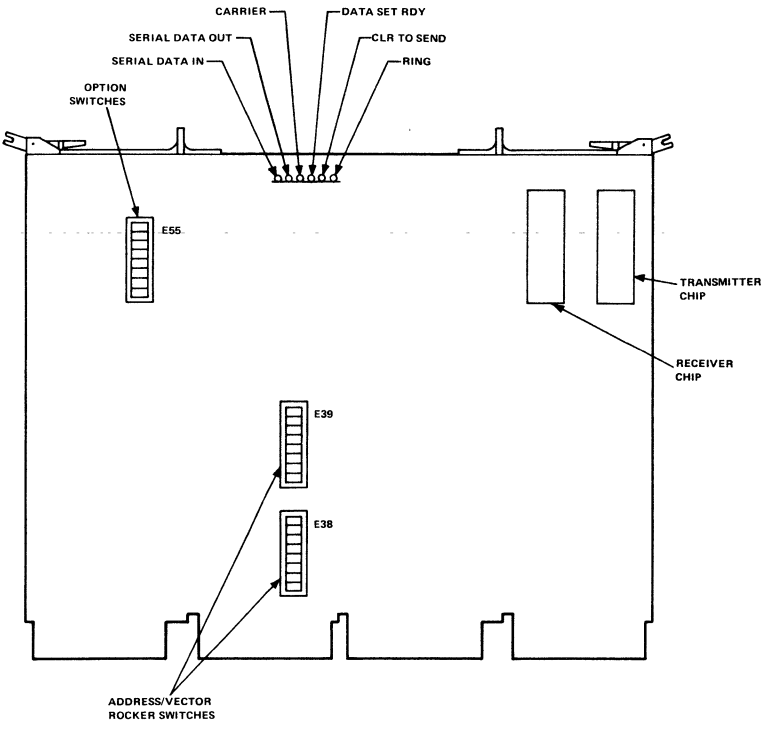


Figure C-53 DUV11 Switch Locations

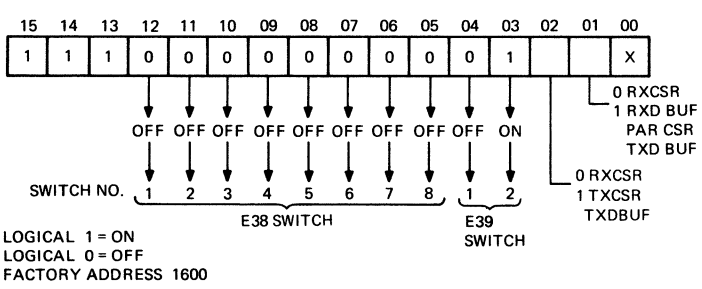


Figure C-54 DUV11 Base Address Selection

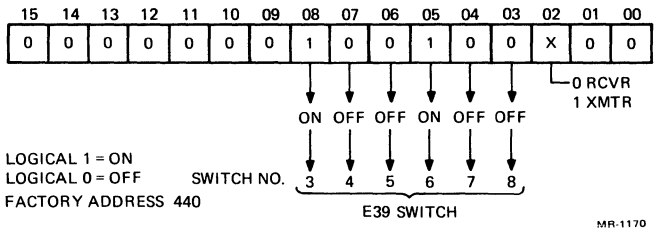


Figure C-55 DUV11 Vector Selection

Specifications

Quad module (M7951)
0.86 A @ +5 Vdc
0.32 A @ +12 Vdc
1.0 ac bus load
1.0 dc bus load
1 × 4 distribution panel insert

Related Documentation

DUV11 Line Interface Technical Manual (EK-DUV11-TM)

Field Maintenance Print Set (MP-00297)

DZV11
DZV11-CP (FOR MICRO/PDP-11)

4-LINE ASYNCHRONOUS SERIAL LINE MULTIPLEXER

Function

Interfaces four asynchronous serial lines to the LSI-11 bus. Incoming data from all four lines is stored in a 64-character first-in, first-out (FIFO) buffer. I/O is under program control; receiver interrupts are generated when the buffer has 1 or 16 characters (as selected by software) and transmitter interrupts are generated when a line is ready to transmit. The buffer can be emptied under program control by repeatedly executing a MOV from the Read Buffer. The ability to access four lines through a single set of device registers reduces the software overhead per character.

Data rates of 50–9600 baud are selected under program control for each line.

Modem Carrier Detect and Ring signals are monitored, and the Data Terminal Ready signal is controlled through the DZV11 registers. The interface standard is RS-232C. See Figures C-56, C-57, and C-58.

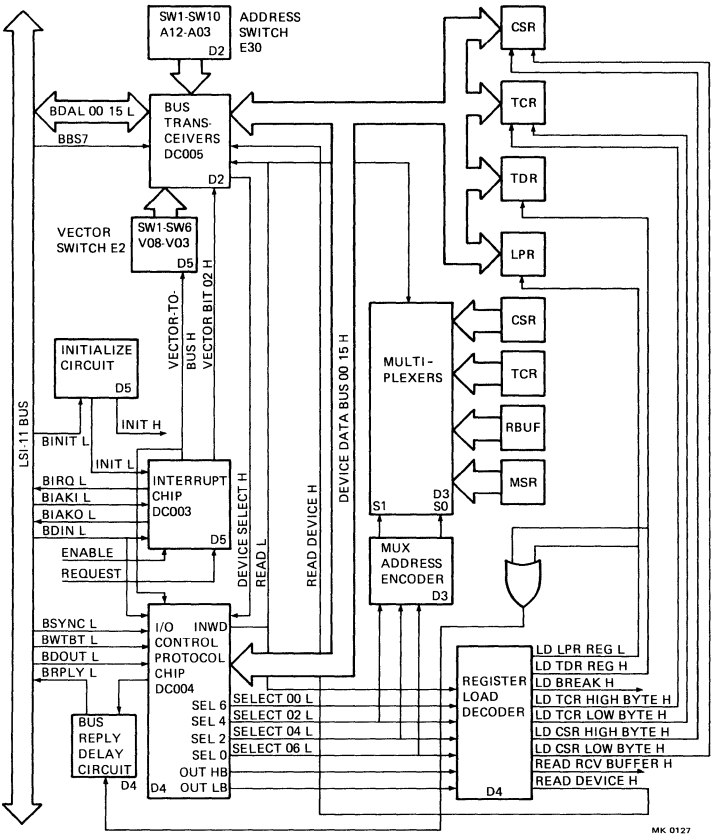


Figure C-56 DZV11 Block Diagram

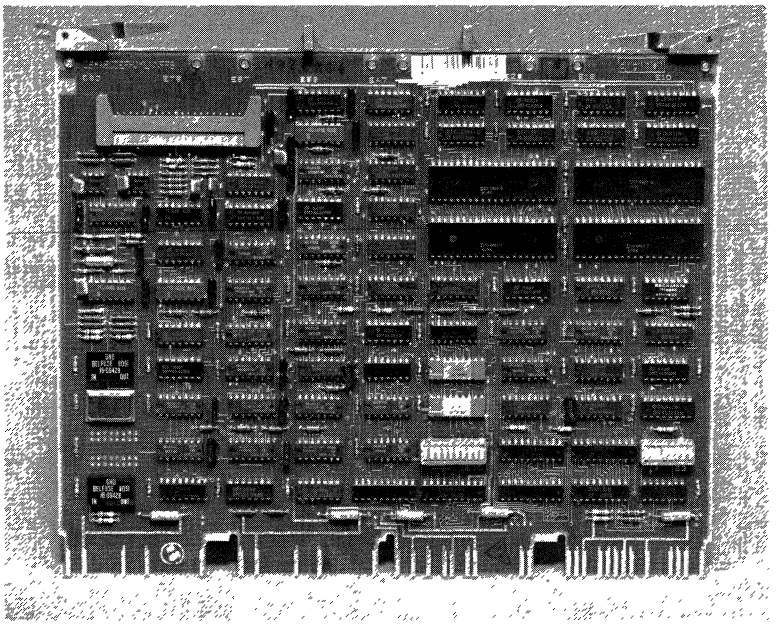


Figure C-57 DZV11

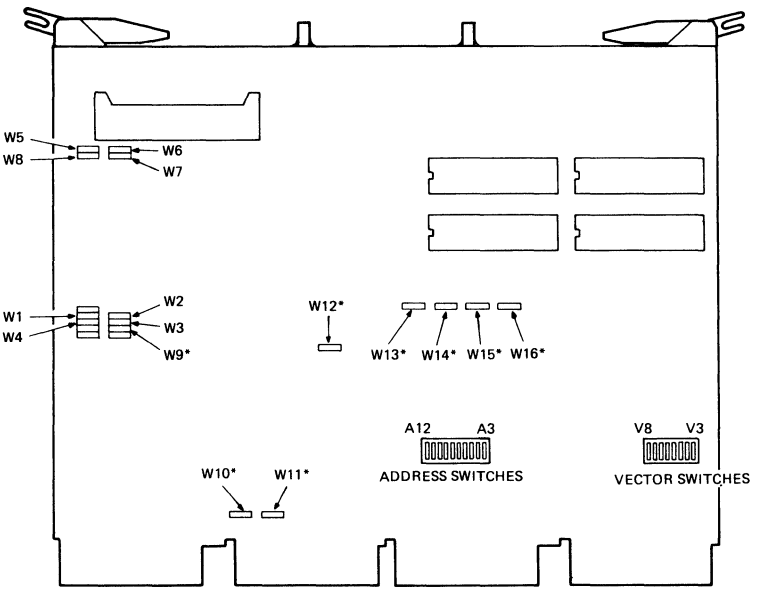
Programming Interface

The device base address is set using switches A12-A3. See Figure C-59.

CSR (R/W)	Base address	Control/Status Register
RBUF (R)	Base address +2	Receiver Buffer
LPR (W)	Base address +2	Line Parameter
TCR (R/W)	Base address +4	Transmitter Control
MSR (R)	Base address +6	Modem Status
TDR (W)	Base address +6	Transmit Data

The interrupt vector is selected with switches V8–V3. See Figure C-60.

The floating address and floating vector conventions are used. See Appendix D.



*NOTES

JUMPERS W9, W12, W13, W14, W15, AND W16 ARE REMOVED ONLY FOR MANUFACTURING TESTS THEY SHOULD NOT BE REMOVED IN THE FIELD.

JUMPERS W10 AND W11 MUST REMAIN INSTALLED WHEN THE MODULE IS USED IN A BACKPLANE THAT SUPPLIES LSI-11 BUS SIGNALS TO THE C AND D CONNECTORS OF THE DZV11 (SUCH AS THE H9270). WHEN THE MODULE IS USED IN A BACKPLANE THAT INTERCONNECTS THE C AND D SECTIONS TO AN ADJACENT MODULE, JUMPERS W10 AND W11 MUST BE REMOVED

MR 24117

Figure C-58 DZV11 Switch Locations

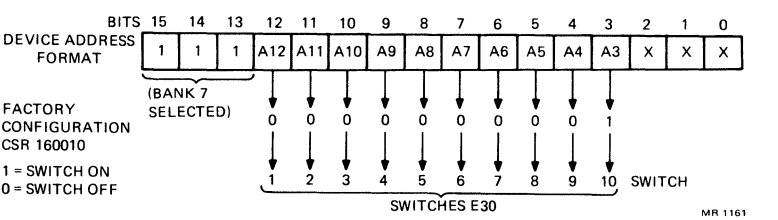


Figure C-59 DZV11 Base Address Selection

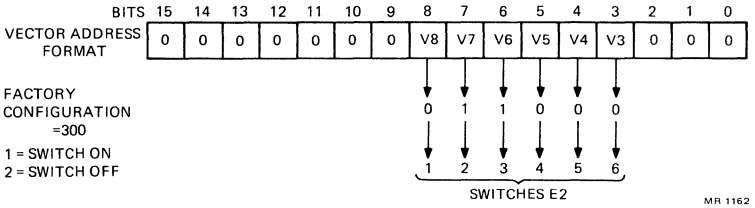


Figure C-60 DZV11 Vector Selection

Physical Interface

DZV11-CP has four 25-pin RS-232C connectors at the distribution panel.

For modem connection, use BC22E or BC22B cables. For local null modem connection, use BC22D or BC22A cables.

Specifications

Quad module (M7957)
1.15 A @ 5 Vdc
0.40 A @ 12 Vdc
4.1 ac bus loads
1 dc bus load
2 × 3 distribution panel insert

Related Documentation

DZV11 Asynchronous Multiplexer Technical Manual (EK-DZV11-TM)

Field Maintenance Print Set (MP-00462)

FPF11-A FLOATING-POINT PROCESSOR

Function

Enhances floating-point performance of F-11-based MICRO/PDP-11 system. Executes all 46 arithmetic operations of the PDP-11 floating-point instruction set. The dedicated 64-bit-wide data path and variable-length microcycle optimize performance. FPF11 monitors the microinstruction flow in the F-11 microprocessor and assumes control over the CPU when a floating-point instruction is fetched. See Figures C61, C-62, and C-63.

Programming Interface

FPF11 connects directly to the F-11 microinstruction bus, and requires no special programming.

Physical Interface

For use in a MICRO/PDP-11 system, jumpers must be configured as follows:

W1, W2, W4, W5, W6, W9, W10	Installed
W3, W7, W8, W11, W12	Removed

FPF11 mounts immediately below the CPU module. A flat I/O conductor cable connects J1 on the FPF11 to the floating-point socket in the CPU module.

Specifications

Quad module (M8188)
5.5 A @ +5 Vdc
No bus loading
No distribution panel insert required

Related Documentation

FPF11 Floating-Point Processor Technical Manual (EK-FPF11-TM)

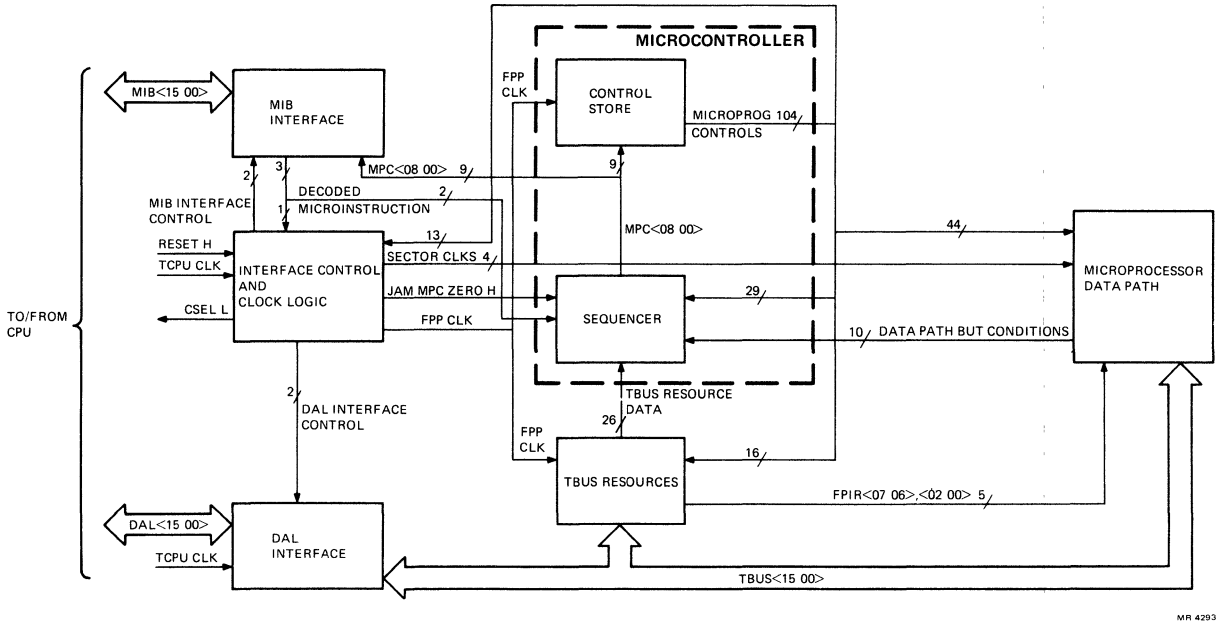


Figure C-61 FPF11 Block Diagram

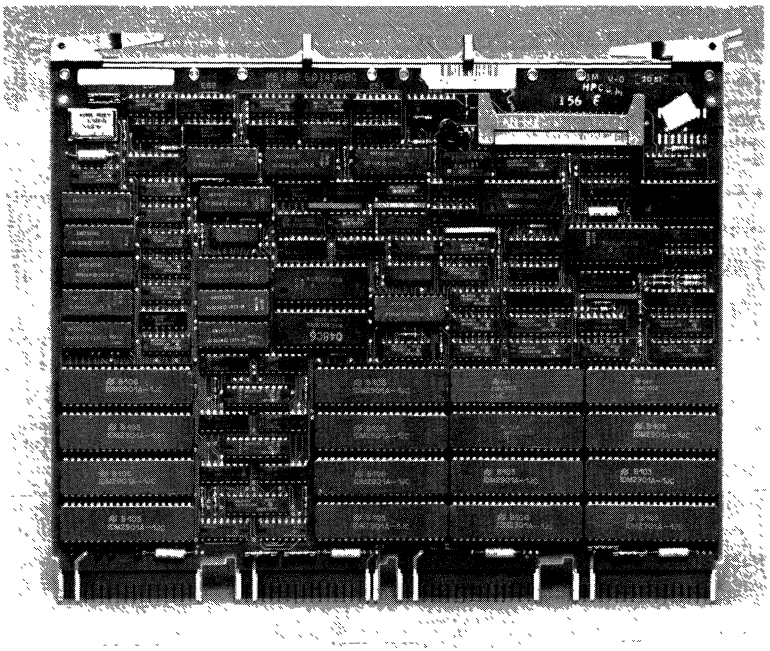


Figure C-62 FPF11

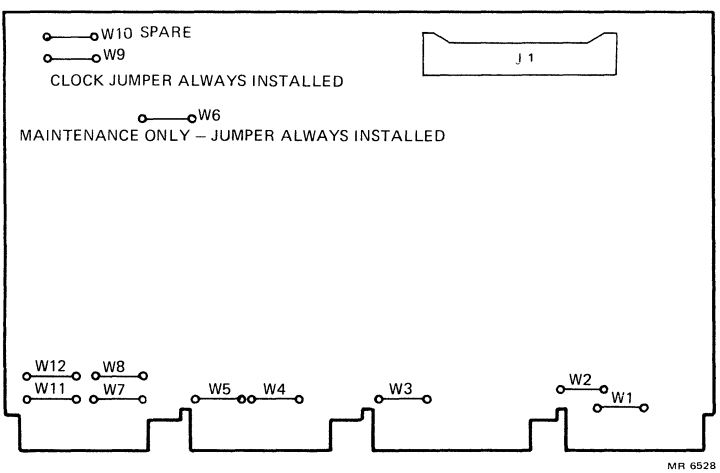


Figure C-63 FPF11 Jumper Locations

IBV11-A **INSTRUMENT**
IBV11-P (FOR MICRO/PDP-11) **BUS INTERFACE**

Function

Interfaces the LSI-11 bus with the instrument bus described in IEEE Standard 488-1975 "Digital Interface for Programmable Instrumentation." Designed to be the only controller on the IEEE bus. IBV11 supports the following subsets of the standard: SH1, AH2, TS, TE5, LE3, SR1, RL1, PP2, DC1, C1-C4.

Program control I/O: interrupts are generated when another device on the bus requests service, when the IBV11 is ready to transmit or to receive, and on an error condition. See Figures C-64, C-65, and C-66.

Programming Interface

The device addresses are set through switches 1–10 in switch pack S2. See Figure C-67.

IBS (R/W)	Base address	Instrument Bus Status Register
IBD (R/W)	Base address +2	Instrument Bus Data Registr

The standard base address is 17 760 150_8.

Vector addresses are set through switches 1–8 in switch pack 51. See Figure C-68.

Error	Vector address
Service request	Vector address +4
Command and talker	Vector address +8
Listener	Vector address +12

The standard vector address is 420_8.

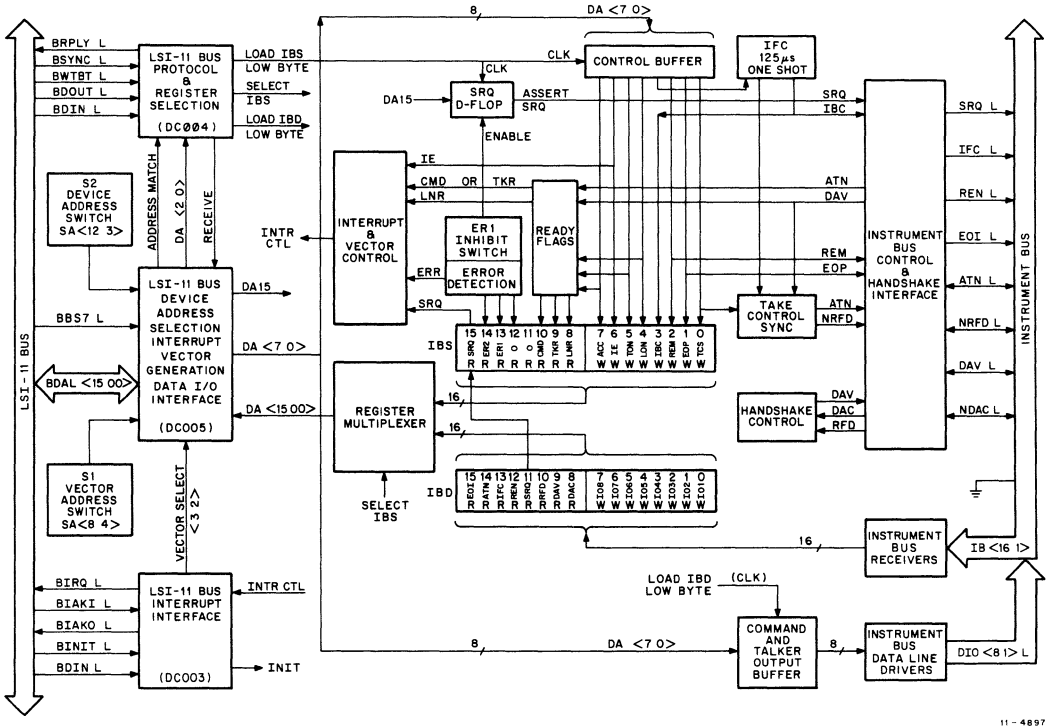


Figure C-64 IBV11 Block Diagram

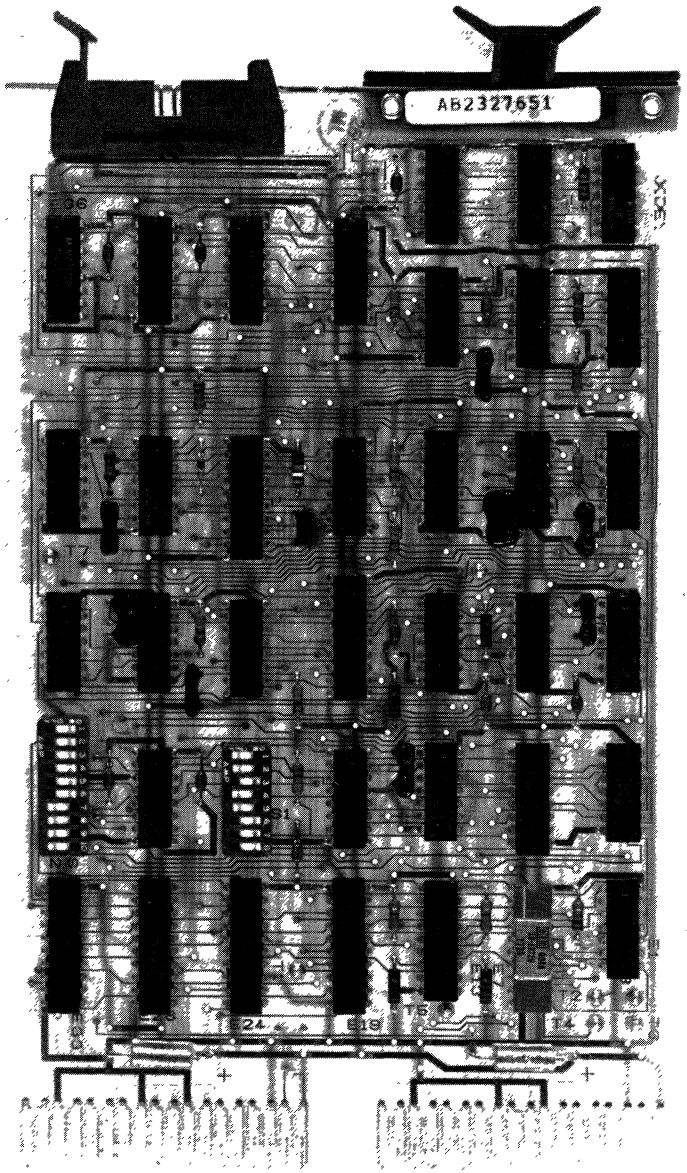


Figure C-65 IBV11

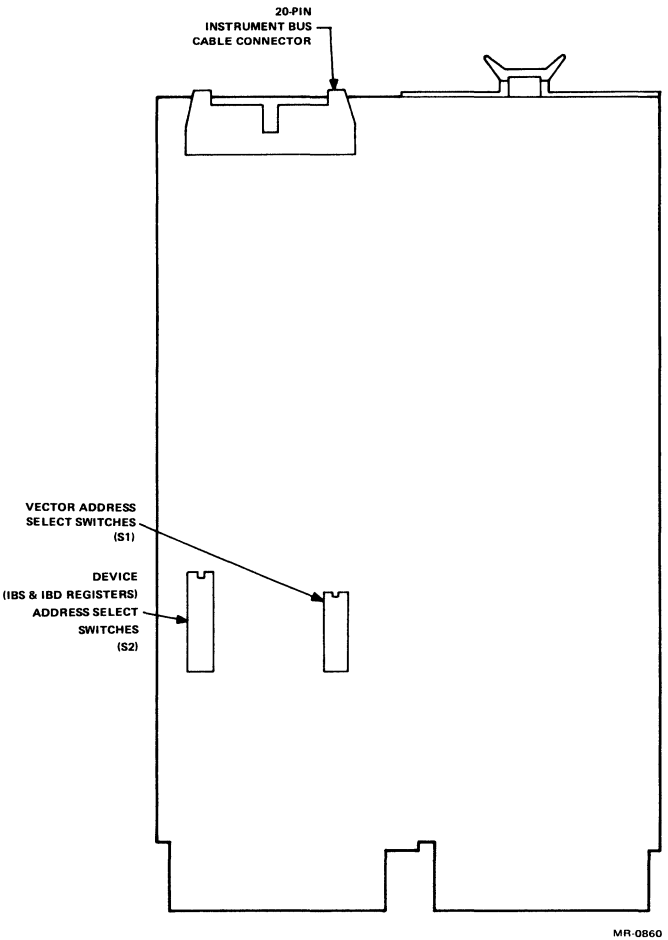


Figure C-66 IBV11 Switch Locations

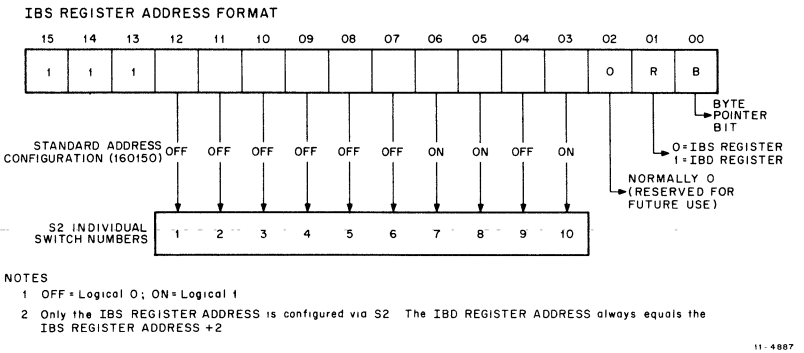


Figure C-67 IBV11 Base Address Selection

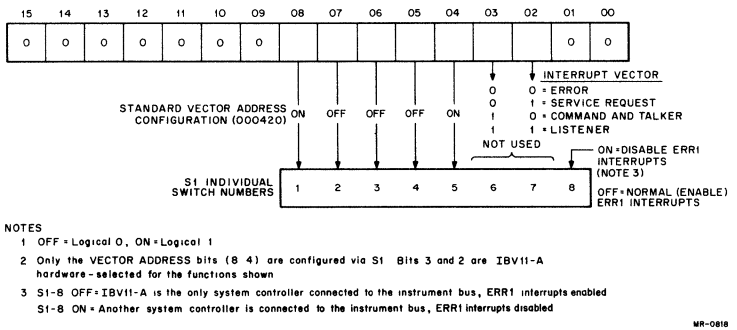


Figure C-68 IBV11 Vector Selection

Physical Interface

The module connector is a 20-pin male connector. IBV11-P repeats this connector at the distribution panel. Connection to the first instrument uses a BN11A cable. Subsequent connections use BNO1A. See Figure C-69.

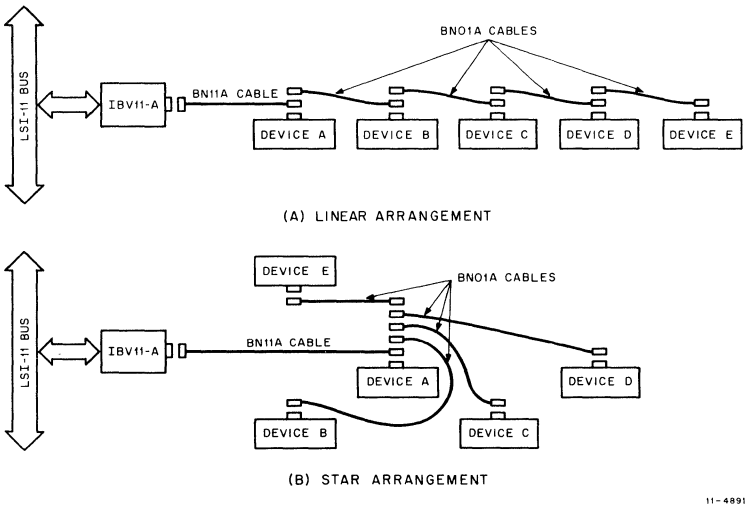


Figure C-69 IBV11 Device Interconnection

Specifications

Double module (M7954)
0.8A @ +5 Vdc
1.77 ac bus loads
1.0 dc bus load
1 × 4 distribution panel insert

Related Documentation

IBV11-A LSI-11/Instrument Bus Interface User's Manual (EK-IBV11-TM)

Digital Interface for Programmable Instrumentation (IEEE Std. 488-1975)

Field Maintenance Print Set (MP-00274)

KWV11-C **PROGRAMMABLE**
KWV11-CP (FOR MICRO/PDP-11) **REAL-TIME CLOCK**

Function
Count or time from internal crystal-controlled frequencies, external signals, or from line frequency.

Description
Four operating modes:

1. Generate a fixed interval based on selected clock rate and count.
2. Generate a fixed frequency pulse train.
3. Record the time of, or count external events. Two events may be monitored relative to each other.
4. Same as item 3, but reset the counter each time the second event occurs.

See Figure C-70.

The KWV11-C has two Schmitt triggers that condition the input waveforms to a form needed by the user. Both can be adjusted to trigger any level in the ∓12 V range (or at TTL fixed levels) and on either the positive or negative slope of the input signal. Each Schmitt trigger has three switches and a potenti ometer. See Figures C-71, C-72, and C-73. The use of these switches and potentiometers is described in the following table.

1: ST1 firing level set by ST1 level adjust pot
2: ST1 fires at TT2 fixed level
3: ST2 firing level set by ST2 level adjust pot
4: ST2 fires at TTL fixed level
5: ST1 firing slope: ON - positive, OFF - negative
6: ST2 firing slope: ON - positive, OFF - negative

The Schmitt triggers can also be controlled externally.

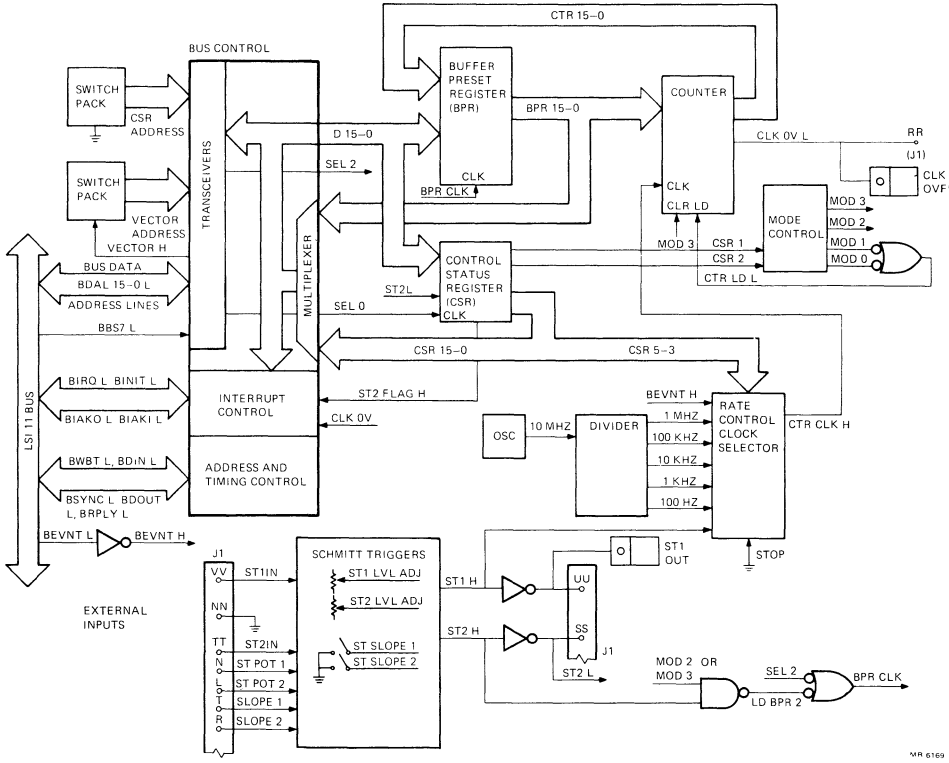


Figure C-70 KWV11 Block Diagram

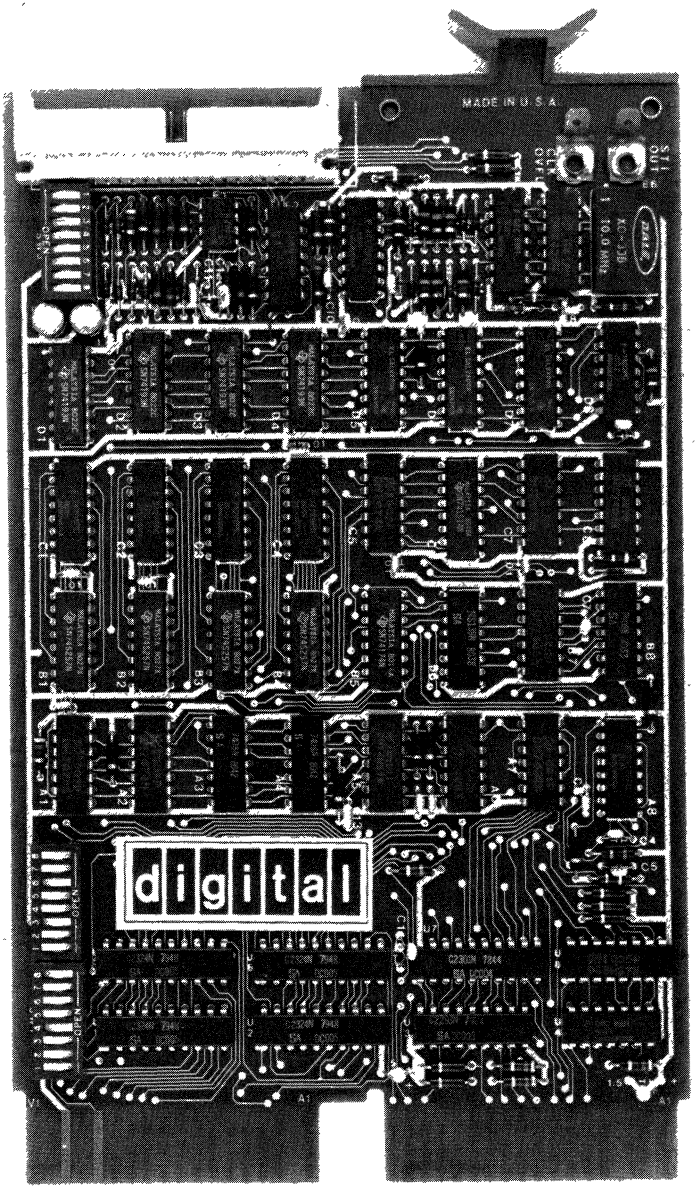


Figure C-71 KWV11

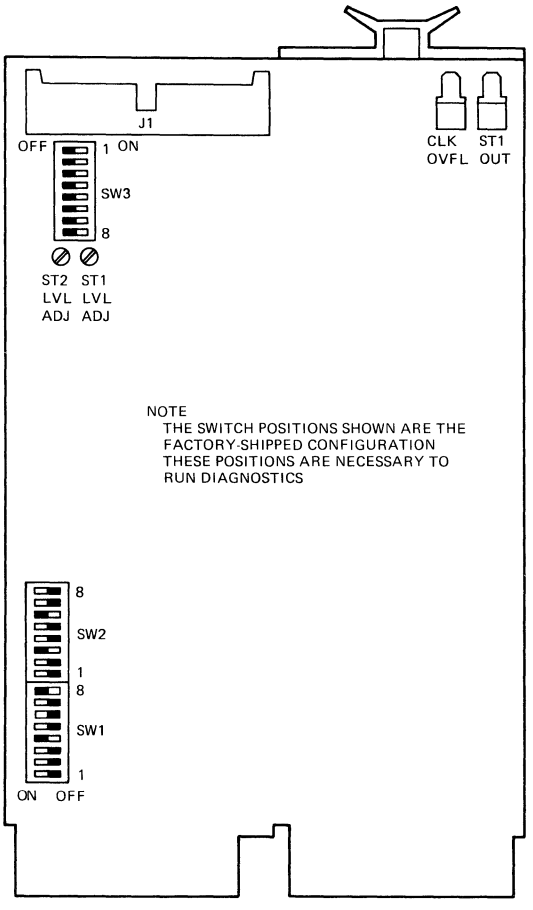


Figure C-72 KWV11 Switch Locations

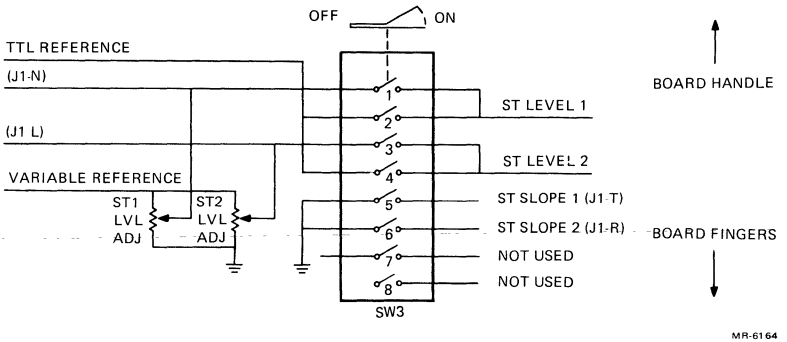


Figure C-73 KWV11 Trigger Reference Levels and Slopes

Programming Interface

The device addresses are set with switches 1–8 of switch pack SW1 and switches 1–2 of switch pack SW2. See Figure C-74.

CSR	(R/W) base address	Control/Status Register
BPR	(R/W) base address +2	Buffer/Preset Register

The standard base address is 17 770 420_8.

The vectors are set through switches 3–8 of switch pack SW2.

CLK OV	Vector address	Clock overflow
ST2	Vector address +4	Schmitt trigger 2

The standard vector address is 440_8. See Figure C-75.

Physical Interface

J1 is a 40-pin male connector. KWV11-CP repeats this connector at the distribution panel. See Figure C-76.

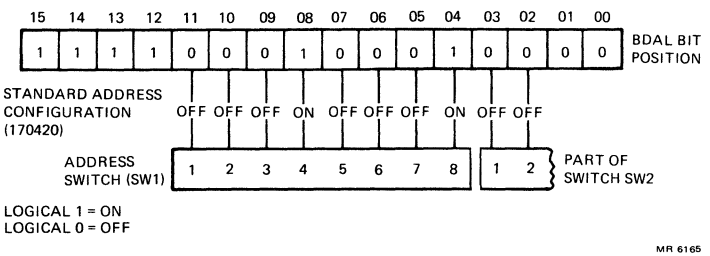


Figure C-74 KWV11 Base Address Selection

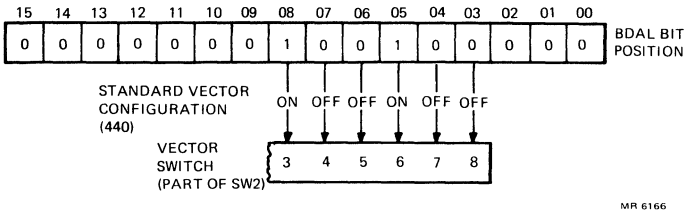


Figure C-75 KWV11 Vector Selection

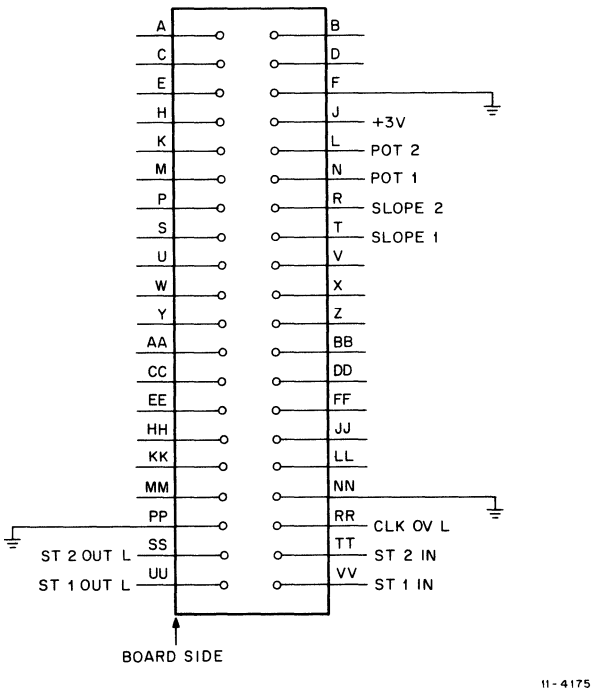


Figure C-76 KWV11 Output Connector

Specifications
Double module (M4002)
2.2 A @ +5 Vdc
.01 A @ +12 Vdc
1.0 ac bus load
1.0 dc bus load
1 x 4 distribution panel insert

Related Documentation

LSI-11 Analog System User's Guide (EK-AXV11-UG)

LPV11 LINE PRINTER
LPV11-CP (FOR MICRO/PDP-11) INTERFACE

Function
Interfaces LP25 and LP26 line printers to the LSI-11 bus. I/O is under program control. See Figures C-77 and C-78.

Programming Interface
The device base address is set using jumpers A12–A3. See Figure C-79.

Control/Status register	Base address
Data Buffer register	Base address +2

The standard base address is 17 777 514_8.

The vector address is set using jumpers V8–V2. See Figure C-80. The standard vector is 200_8.

Physical Interface
LPV11 connects at the distribution panel using a unique connector.

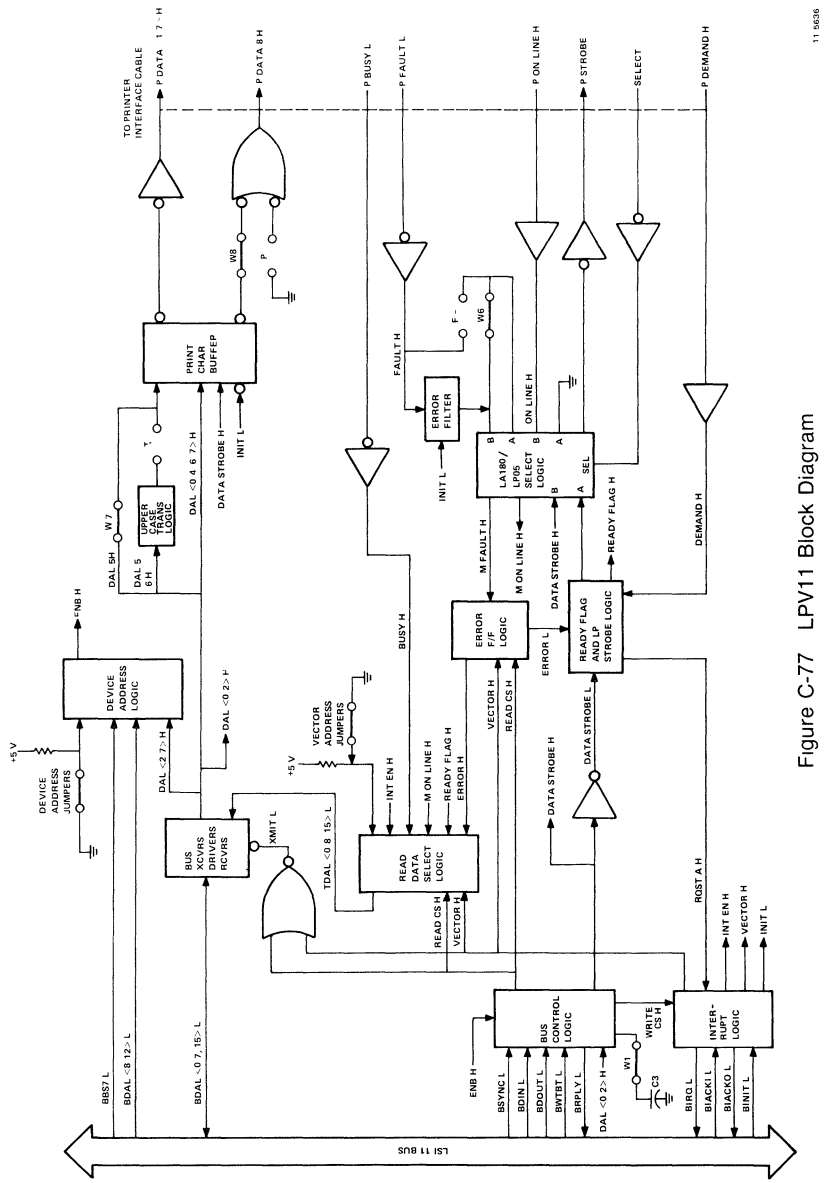
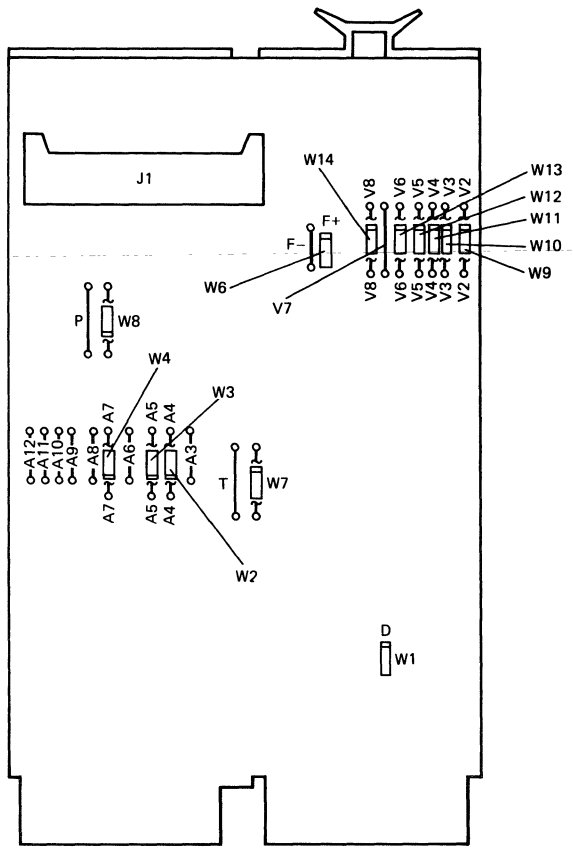


Figure C-77 LPV11 Block Diagram



NOTE:

- ⚲ =JUMPERS BROKEN FOR CLARITY ON THIS FIGURE. THESE WIRE-WRAP JUMPERS WOULD NORMALLY BE USED TO REPLACE PREVIOUSLY REMOVED FACTORY INSTALLED (W) JUMPERS (SHOWN INSTALLED).
- o =WIRE WRAP PIN.

MR-0863

Figure C-78 LPV11 Jumper Locations

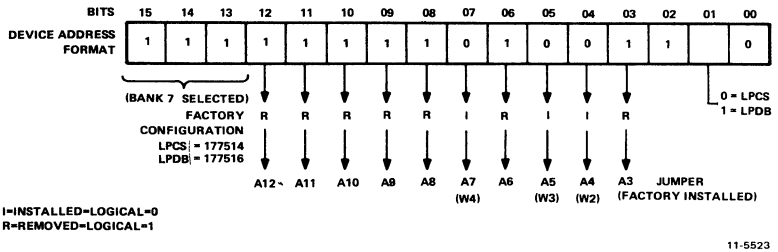


Figure C-79 LPV11 Base Address Selection

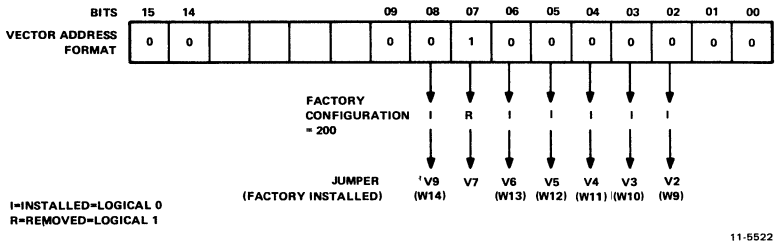


Figure C-80 LPV11 Vector Address Selection

Specifications

Double module (M8027)
0.8 A @ +5 Vdc
1.4 ac bus loads
1.0 dc bus load
1 × 4 distribution panel insert

Related Documentation

LPV11 Line Printer User's Guide (EK-LPV11-OP)

Field Maintenance Print Set (MP-00467)

MCV11-DA (8KB)
MCV11-DC (32KB)

CMOS MEMORY WITH BATTERY BACKUP

Function

Memory module with on-board battery backup to protect data when power is removed.

Typical data retention times are 1180 hours (-DC) and 2647 hours (-DA). Worst case data retention times are 100 hours (-DC) and 333 hours (-DA). See Figures C-81 and C-82.

Programming Interface

The Module Starting Address is set by jumpers between pins L,M,N,P and ground (pin R) and between pins A,B,C,D,E and ground (pin F). Each pin corresponds to one bit of the 22-bit address. When a jumper is in, the corresponding bit is 1. When a ju mper is out, the corresponding bit is 0. See Figure C-83.

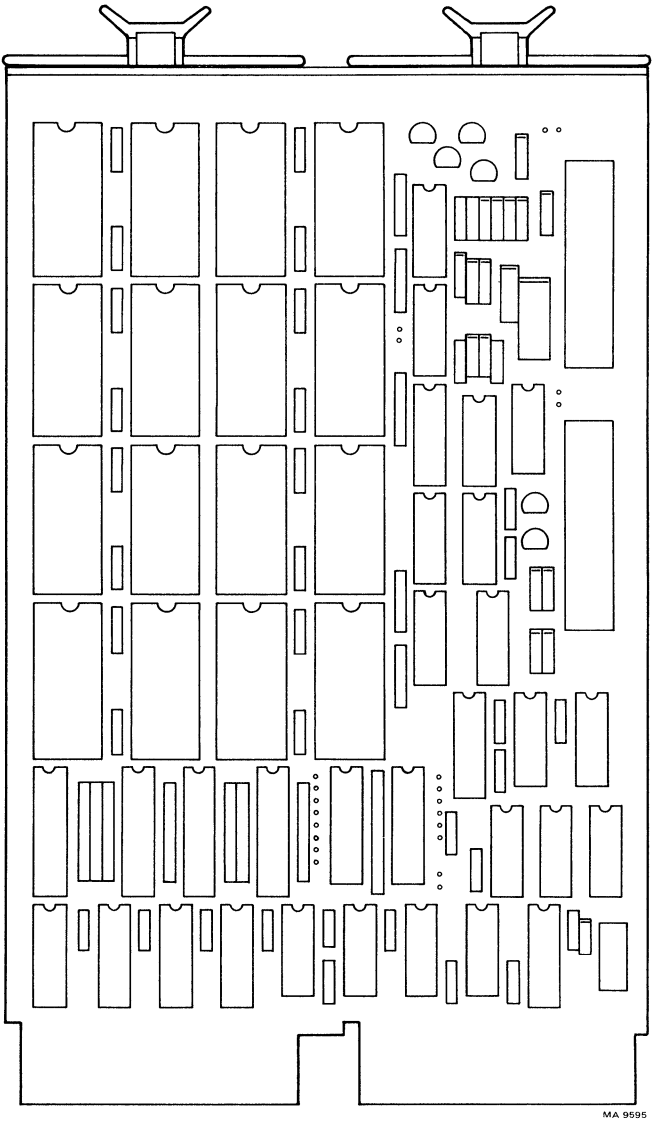


Figure C-81 MCV11 Jumper Locations

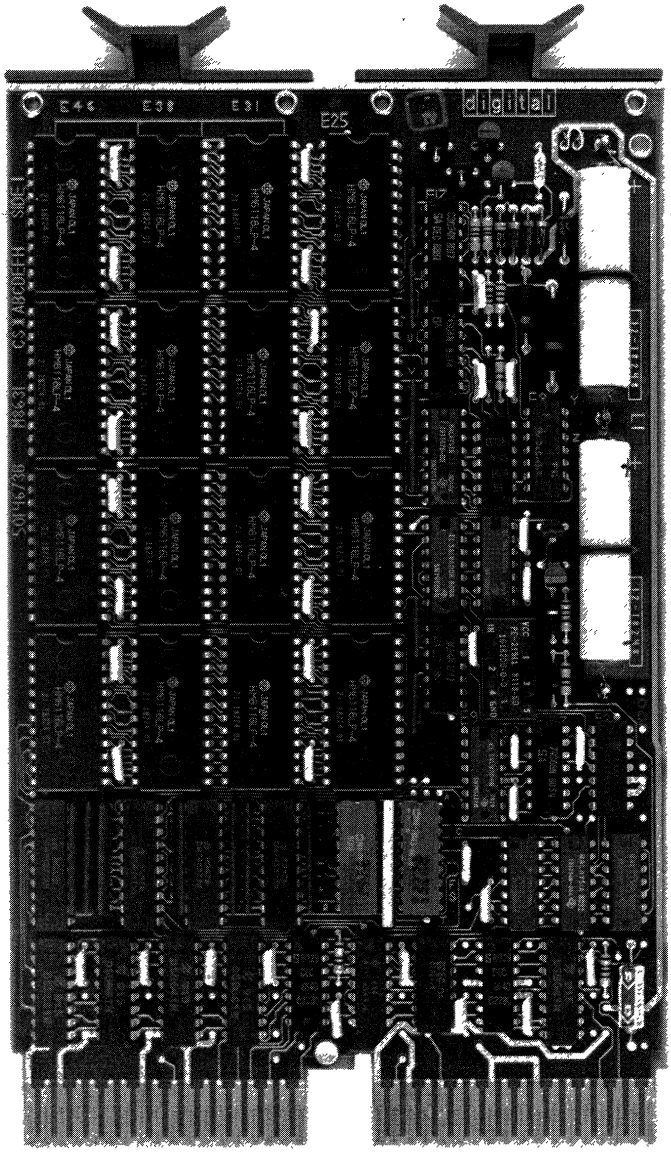


Figure C-82 MCV11

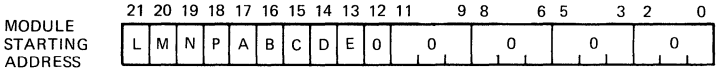


Figure C-83 MCV11 Module Starting Address

Specification

Double module (M8631)
1.2 A @ +5 Vdc
2 ac bus loads
1 dc bus load
No distribution panel insert required

Related Documentation

MCV11-D User's Guide (EK-MCV1D-UG)
MCV11-D Reference Card (EK-MCV1D-RC)
Field Maintenance Print Set (MP-01309)

MSV11-LF **256 KB PARITY MEMORY**

Function

Memory module with parity. See Figures C-84 and C-85.

Programming Interface

The Module Starting Address is set by jumpers between pins P, N, M, L and ground (pin K) and between pins Z, Y, X, W, V and ground (pin V). Each pin corresponds to a bit of the 22-bit address. When a jumper is in, the corresponding bit is 1, when the jumper is out, the corresponding bit is 0. See Figure C-86.

The Control/Status Register address is set by jumpers between pins A, B, C and ground (pin E). Each pin corresponds to a bit in the 22-bit address. When a jumper is in, the corresponding bit is 1; when the jumper is out, the corresponding bit is 0. See Figure C-87.

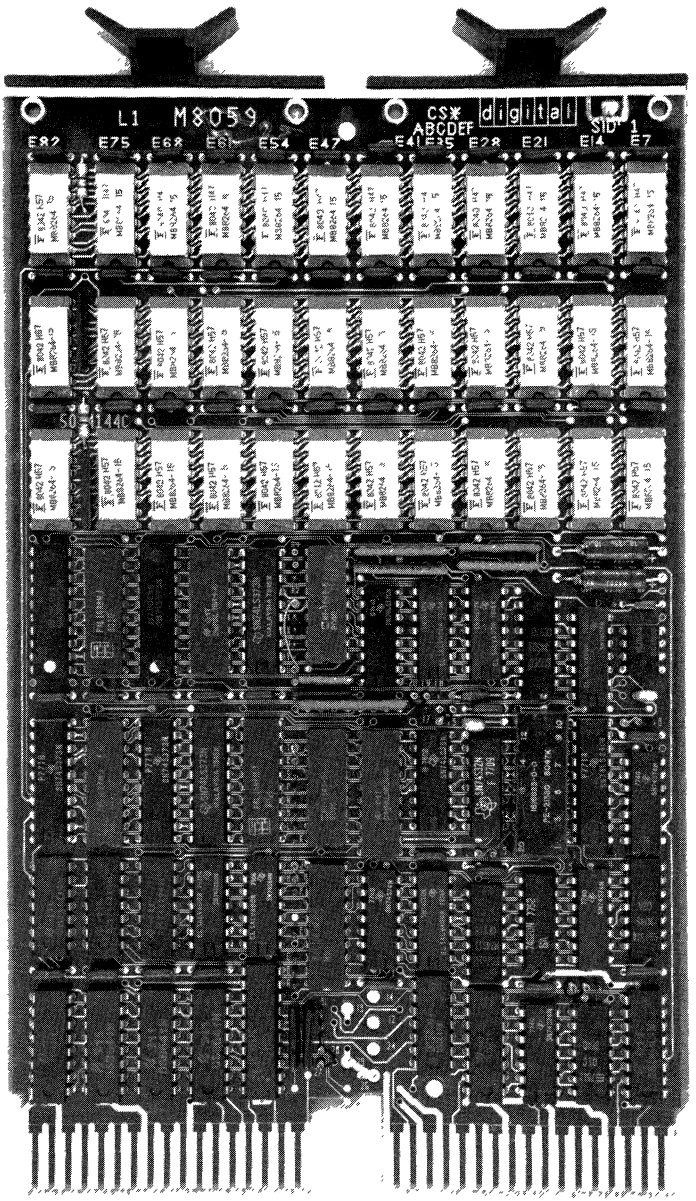


Figure C-84 MSV11-L

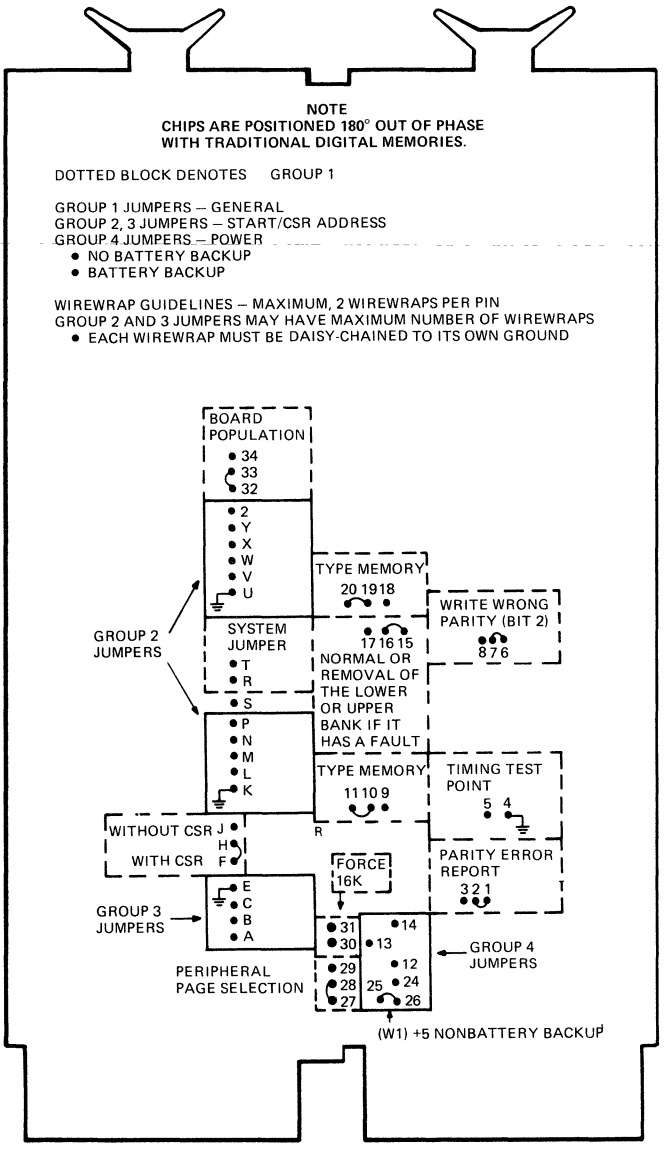


Figure C-85 MSV11-L Jumper Locations

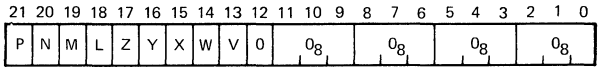


Figure C-86 MSV11-L Module Starting Address

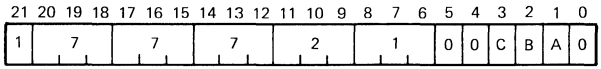


Figure C-87 MSV11-L Vector Address

Specifications

Double module (M8059)
3.9 A @ +5 Vdc
2 ac bus loads
1 dc bus load
No distribution panel insert required

Related Documentation

MSV11-L User's Guide (EK-MSV0L-UG)

Field Maintenance Print Set (MP-01238)

MSV11-PK (256KB) **PARITY MEMORY**
MSV11-PL (512KB)

Function

Memory modules with parity. Support block-mode DMA transfers for increased throughput. See Figures C-88 and C-89.

Programming Interface

The Module Starting Address is set with jumpers between plus X, W, V, P, N, M, L, T and ground (pin Y). Each pin corresponds to one bit of the 22-bit address. When a jumper is in, the corresponding bit is 1; when the jumper is out, the corresponding bit is 0. See Figure C-90.

The Control/Status Register address is set with jumpers between pins A, B, C, D and ground (pin E). Each pin corresponds to one bit in a 22-bit address in the range 177 721 00-177 721 36_8. When a jumper is in, the corresponding b it is 1; when the jumper is out, the corresponding bit is 0. See Figure C-91.

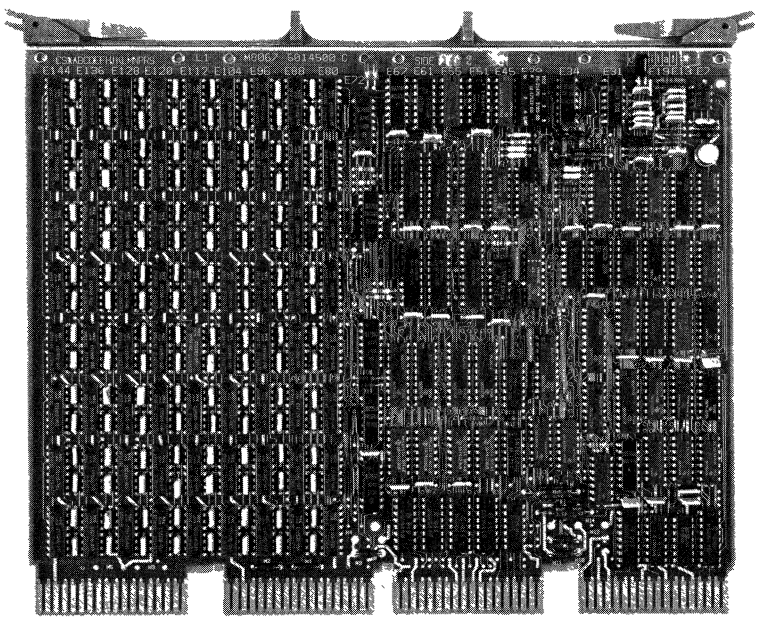


Figure C-88 MSV11-P

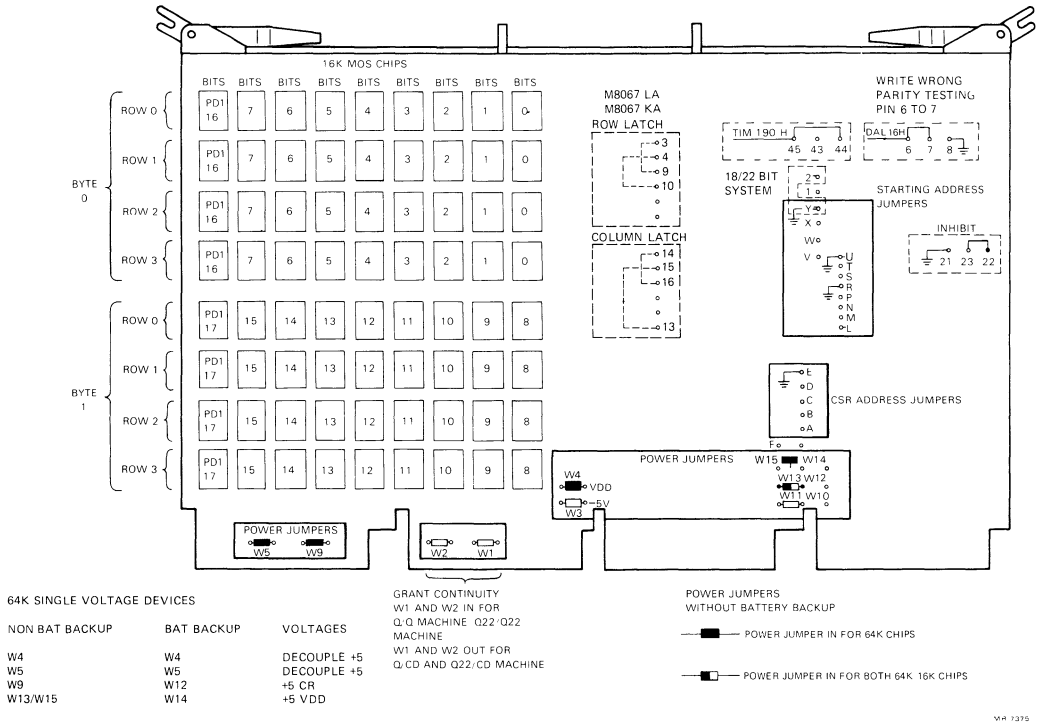


Figure C-89 MSV11-P Jumper Locations

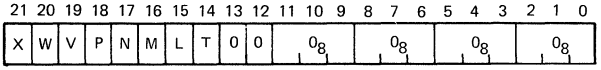


Figure C-90 MSV11-P Module Starting Address

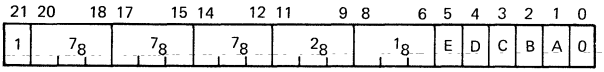


Figure C-91 MSV11-P Vector Address

Specifications

Quad module (M8067)
3.45 A @ +5 Vdc (MSV11-PK)
3.60 A @ +5 Vdc (MSV11-PL)
2 ac bus loads
1 dc bus load
No distribution panel space required

Related Documentation

MSV11-P User's Guide (EK-MSV0P-UG)

Field Maintenance Print Set (MP-01239)

RLV12 **RL01/RL02 DISK CONTROLLER**

Function

Interface 1-4 RL01 or RL02 disk drives to the LSI-11 bus. I/O is DMA. RLV12 performs a cyclic redundancy check on data and headers. Memory parity is checked, and the current command to the RLV12 is aborted when an error is detected. RL02 (10MB) and RL01 (5MB) disk drives are 10.5-inch high rack-mount devices with top-loading removable disk cartridges. See Figures C-92, C-93, and C-94.

Programming Interface

The device base address is set with pins M21–M12; grounding a pin (to pin M22) sets the corresponding address bit to one. See Figure C-95.

Control/Status register	Base address
Bus address register	Base address +2
Disk address register	Base address +4
Multipurpose register	Base address +6
Bus address extension	Base address +8

The standard base address is 17 774 400_8.

The interrupt vector is set using pins M10–M4; grounding a pin (to pin M3) sets the corresponding bit to one. See Figure C-96.

The standard vector address is 160_8.

Physical Connection

RLV12 connects at the distribution panel with a unique connector.

A cable connects the first RL01/RL02 drive to the RLV12. The second drive connects to the first.

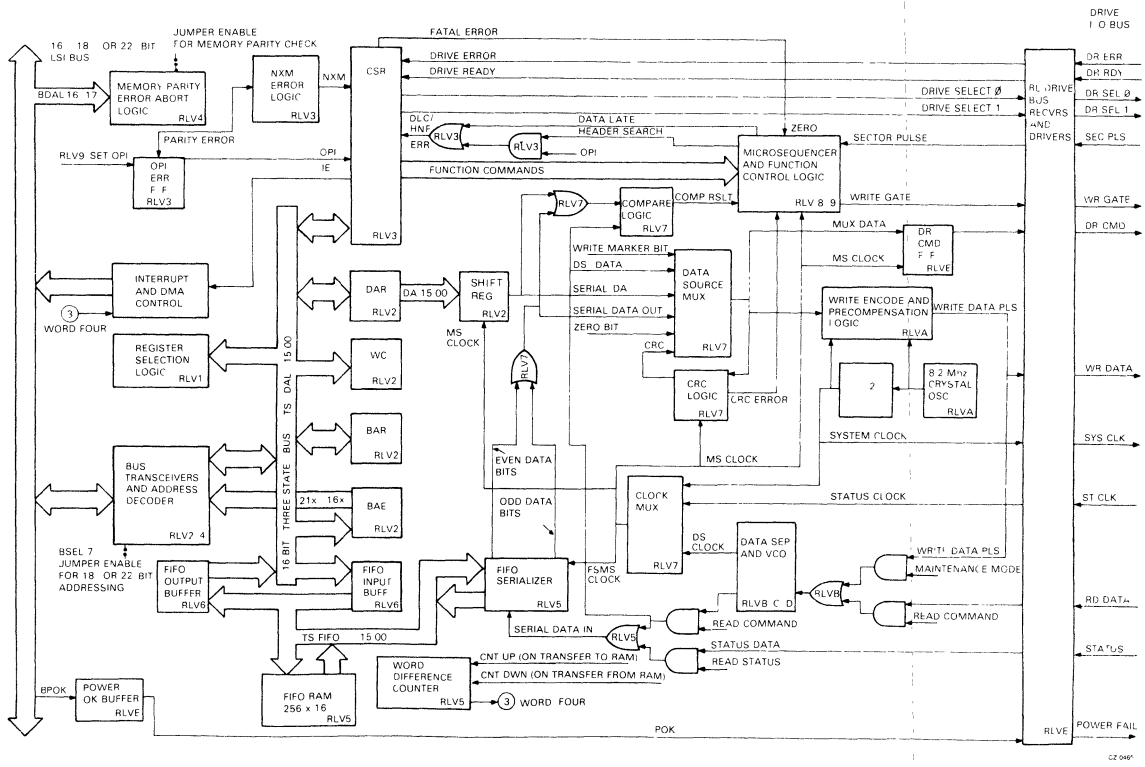


Figure C-92 RLV12 Block Diagram

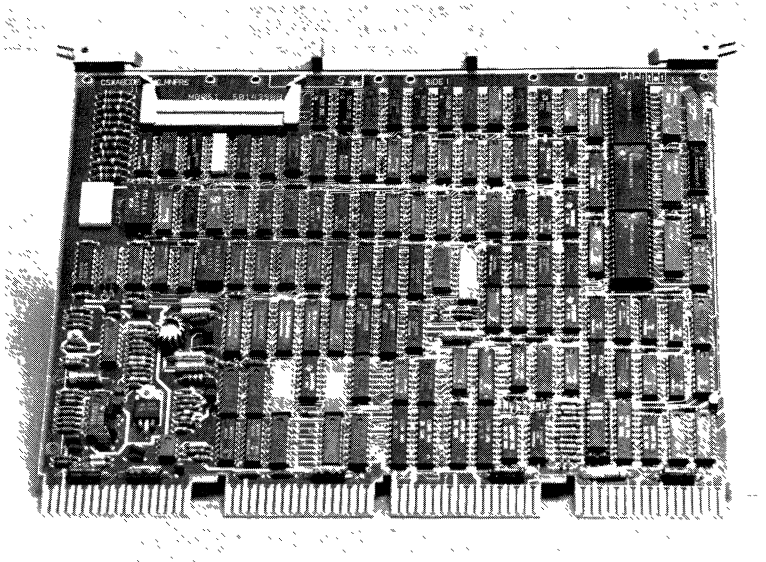
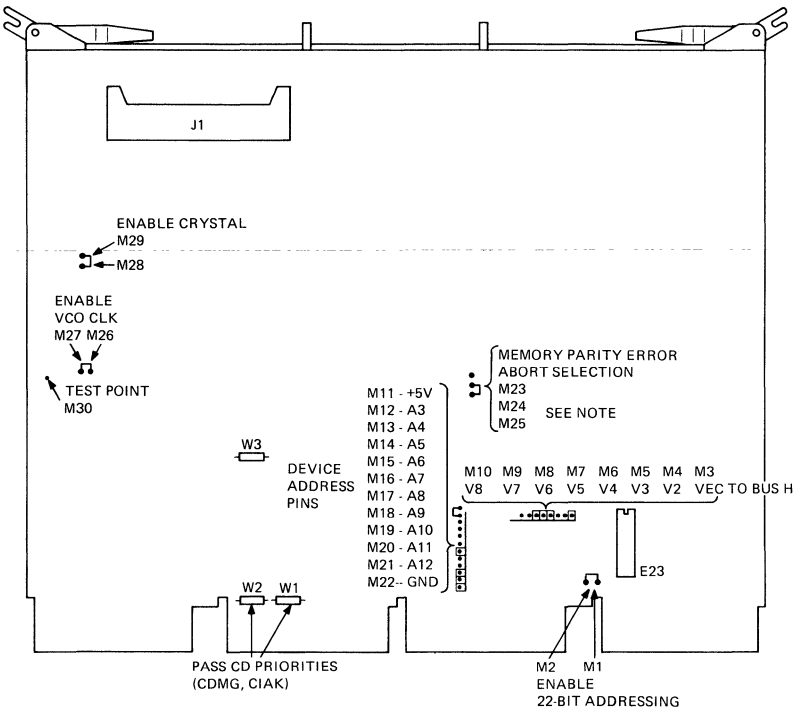


Figure C-93 RLV12



NOTE
THE MEMORY PARITY ERROR ABORT FEATURE IS AVAILABLE FOR USE WITH MEMORIES THAT HAVE PARITY ERROR CHECKING
THIS FEATURE DOES NOT HAVE TO BE DISABLED FOR MEMORIES THAT DO NOT HAVE PARITY ERROR

CHECKING THE PINS ARE CONNECTED AS FOLLOWS

CONNECTION	FUNCTION
M23 - M24	NO PARITY
M24 - M25	PARITY ERROR ABORT

Figure C-94 RLV12 Jumper Locations

Specifications

Quad module (M8061)
5.0 A @ +5 Vdc
0.1 A @ +12 Vdc
2.7 ac bus loads
1.0 dc bus loads
1 × 4 distribution panel insert

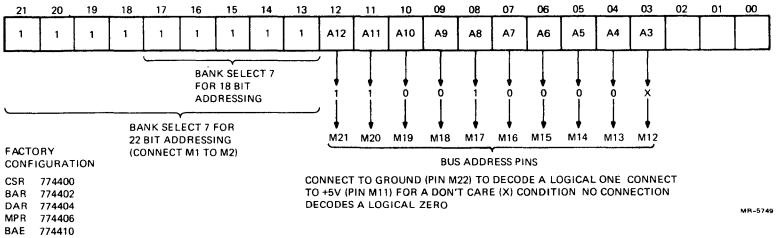


Figure C-95 RLV12 Base Address Selection

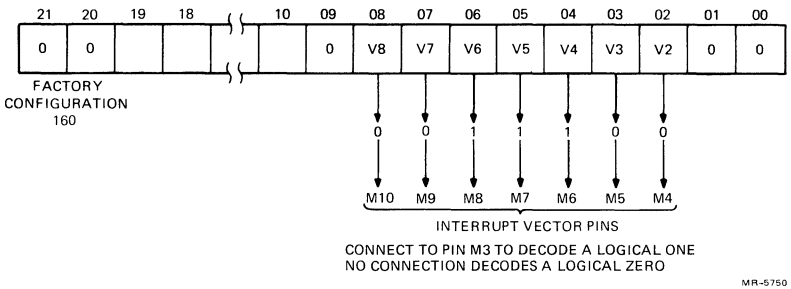


Figure C-96 RLV12 Vector Address Selection

Related Documentation

RLV12 Disk Controller User's Guide (EK-RLV12-UG)

RL01/RL02 User's Guide (EK-RL012-UG)

RL01/RL02 Pocket Service Guide (EK-RL012-PG)

RQDX1 **RD50/RX50 CONTROLLER**

Function

Interfaces RD51 10MB fixed disk drives and RX50 dual 400KB diskette drives to the LSI-11 bus. I/O is block-mode DMA. Communication with the drives uses Mass Storage Control Protocol (MSCP). RQDX1 can control up to four logical units: each RD51 counts as one logical unit, and each RX50 as two. See Figures C-97 and C-98.

Programming Interface

The device base address is set using jumpers A2–A12. See Figure C-99.

The standard base address is 17772150_8.

The interrupt vector is program-selectable, in the range $4\text{-}774_8$. The standard vector address is 154_8.

Physical Interface

RQDX1 connects to RD51 and RX50 drives in the MICRO/PDP-11 system chassis through internal cabling and connectors.

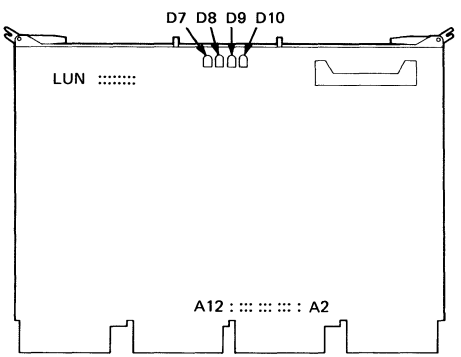


Figure C-97 RQDX1 Jumper Locations

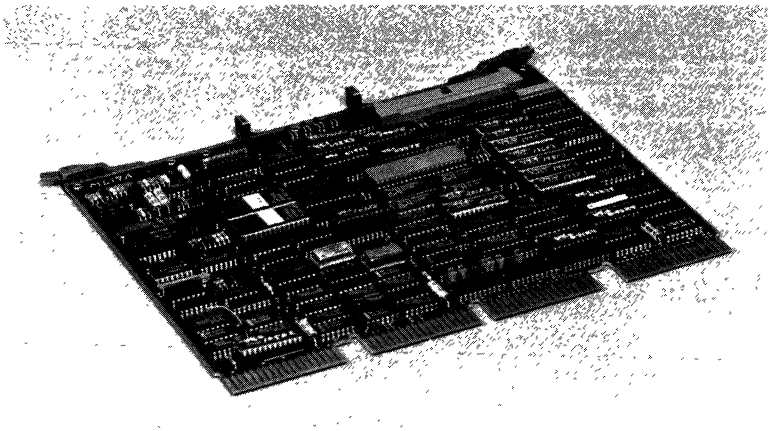


Figure C-98 RQDX1

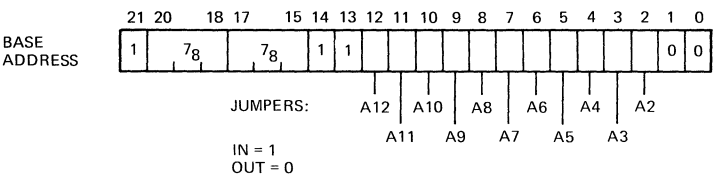


Figure C-99 RQDX1 Base Address Selection

Specifications

Quad module (M8638)
5.8 A @ +5 Vdc
.03 A @ +12 Vdc
2 ac bus loads
1 dc bus load

Related Documentation

RQDX1 Controller Module User's Guide (EK-RQDX1-UG)

TSV05 **MAGNETIC TAPE SUBSYSTEM CONTROLLER**

Function

The TSV05 magnetic tape subsystem reads and writes 1600 bpi, 9-track tape at 25 inches/second in TS11 compatibility mode, or 100 inches/second with user-provided software.

Tape data is buffered in 3.5K byte of RAM on the controller. Automatic read after write verifies accurate data recording. See Figure C-100.

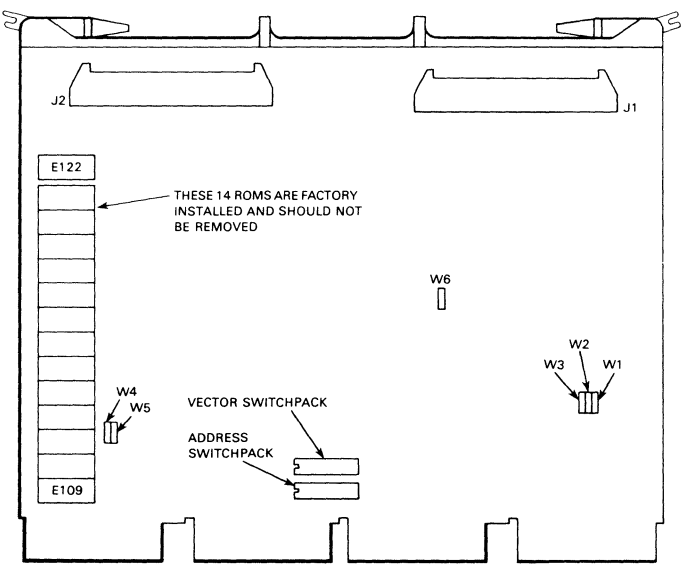


Figure C-100 TSV05 Switch Locations

Programming Interface

The device base address is set with switches E58 <10> and E57 <1:10>. See Figure C-101.

The standard base addresses are:

$17\ 772\ 520_8$ 1st TSV05

$17\ 772\ 524_8$ 2nd TSV05

$17\ 772\ 530_8$ 3rd TSV05

$17\ 772\ 534_8$ 4th TSV05

The vector is set with switch E58 <1:7>.

The standard vector for the first TSV05 in a system is 244_8. Subsequent units have a rank of 37 in floating vector space (Appendix D).

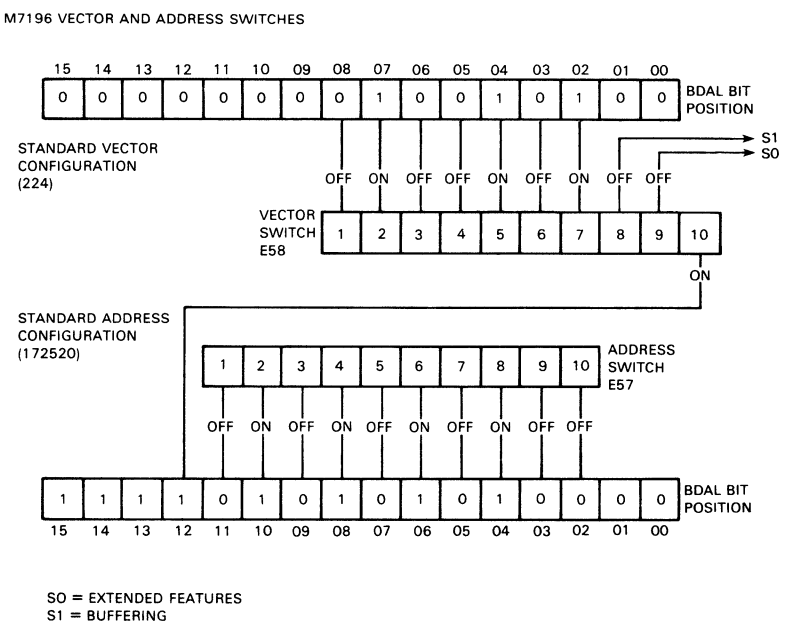


Figure C-101 TSV05 Base Address and Vector Address Selection

Physical Interface

TSV05 connects through two 50-pin connectors at the distribution panel.

Specifications

Quad module (M7196)
6.5 A @ +5 Vdc
3 ac bus loads
1 dc bus load
two 1 × 4 distribution panel inserts

Related Documentation

TSV05 Tape Transport System User's Guide (EK-TSV05-UG)

APPENDIX D
VECTOR AND I/O PAGE ADDRESS ASSIGNMENTS

This Appendix contains a list of all vector and I/O page-address assignments applicable to MICRO/PDP-11 systems (Tables D-1 and D-2). Assignments implemented on MICRO/J-11 but not on the F-11-based MICRO/PDP-11 are included for reference and are indicated by an asterisk (*).

When a device has been assigned specific vector or I/O page addresses, those addresses should be used first. Additional device addresses and vectors should be located in the floating CSR area (17 760 010 - 17 763 776) and the Floating Vector area (300 - 776).

Each device eligible for the floating CSR or floating vector areas is assigned a rank (identified in the lists). The highest ranked device (smallest number) is assigned the first vector address (300) or CSR address (17 760 010), and subsequent devices are assigned addresses in ascending sequence. Refer to the option description in Appendix C to determine how many words are required for a particular device's registers or vectors.

Note: Addresses not listed as assigned in this Appendix may be reserved for DIGITAL use. Refer to the *PDP-11 Architecture Handbook.*

Table D-1 Interrupt and Trap Vector Assignments

Address (octal)	Function
004	Bus time-out and illegal instructions
010	Illegal and reserved instructions
014	BPT instruction and trace trap
020	IOT instruction
024	Power-fail
030	EMT instruction
034	TRAP instruction
060	Console terminal, input
064	Consol terminal, output
100	External event line interrupt
124	DRV11-B parallel interface
160	RLV12 disk controller
200	LPV11 line printer
240*	PIRQ
244	Floating-point error
250	Memory management
264	RXV21 floppy disk
300–777	Floating vectors (rank): DLV11-J (2) DRV11-B (8) DRV11 (9) DLV11-E (14) KWV11 (20) DUV11 (26) DZV11 (27) RLV12 (34) RXV21 (39) DPV11 (43) DMV11 (46)

* Not implemented on F-11-based MICRO/PDP-11; implemented on MICRO/J-11.

Table D-2 I/O Page Addresses

Address (Octal)	Function
17 760 010 - 17 763 776	Floating addresses (rank): DUV11 (4) DZV11 (8) RLV11, 12 (14) RXV21 (18) DPV11 (21) DMV11 (24)
17 764 100 - 17 764 106	DRV11-J#1
17 764 110 - 17 764 116	DRV11-J#2
17 764 120 - 11 764 126	DRV11-J#3
17 767 740 - 17 767 742	DRV11#3
17 767 744 - 17 767 746	DRV11#2

Table D–2 I/O Page Addresses (Cont)

Address (Octal)	Function
17 767 750 - 17 767 752	DRV11#1
17 770 400	ADV11
17 770 420	KWV11-A
17 770 440	AAV11
17 770 420 - 17 777 421	ADV11#2, AXV11#2, KWV11-C
17 770 450	ADV11#1, AXV11#1
17 772 200 - 17 772 216*	Supervisor Mode I Space PDR 0-7
17 772 220 - 17 772 236*	Supervisor Mode D Space PDR 0-7
17 772 240 - 17 772 256*	Supervisor Mode I Space PAR 0-7
17 772 260 - 17 772 276*	Supervisor Mode D Space PAR 0-7
17 772 300 - 17 772 316	Kernel Mode I Space PDR 0-7
17 772 320 - 17 772 336*	Kernel Mode D Space PDR 0-7
17 772 340 - 17 772 356	Kernel Mode I Space PAR 0-7
17 772 360 - 17 772 376*	Kernel Mode D Space PAR 0-7
17 772 410	DRV11-B#1
17 772 420	DRV11-B#2
17 772 430	DRV11-B#3
17 772 516	Memory Management Status Register 3
17 773 000 - 17 773 776	KDF11-B Boot/Diagnostic ROM
17 774 400	RLV11, RLV12
17 775 610 - 17 776 476	31 serial-line units with modem control (DLV11-E)
17 776 500 - 17 776 676	16 serial-line units without modem control (DLV11-J)
17 777 170	RXV21 (RX02 floppy diskette)
17 777 512	Line printer (LPV11)
17 777 520 - 17 777 524	KDR11-B Boot/Diagnostic Registers
17 777 546	Line time clock
17 777 560	Console Terminal
17 777 514	LPV11
17 777 572	Memory Management Status Register 0
17 777 574	Memory Management Status Register 1
17 777 576	Memory Management Status Register 2
17 777 600 - 17 777 616	User Mode I Space PAR 0-7
17 777 620 - 17 777 636*	User Mode D Space PAR 0-7
17 777 640 - 17 777 656	User Mode I Space PDR 0-7
17 777 660 - 17 777 676*	User Mode D Space PDR 0-7
17 777 746*	Cache Control Register
17 777 752*	Hit/Miss Register
17 777 766*	CPU Error Register
17 777 772*	Program Interrupt Request Register (PIR)
17 777 776	Processor Status Word (PSW)

* Not implemented in F-11-based MICRO/PDP-11; implemented on MICRO/J-11.

APPENDIX E
LSI-11 BUS TECHNICAL SPECIFICATIONS

The LSI-11 Bus is the low-end member of DIGITAL's bus family. All DIGITAL microcomputers use the LSI-11 Bus. However, in order to use the 22-bit addressing capabilities of the LSI-11/23, the MICRO/PDP-11, and the PDP-11/23-PLUS, the extended LSI-11 Bus is required.

The LSI-11 Bus consists of 42 bidirectional and 2 unidirectional signal lines. These form the lines along which the processor, memory, and I/O devices communicate with each other.

Addresses, data, and control information are sent along these signal lines, some of which contain time-multiplexed information. The lines are divided as follows:

- Sixteen multiplexed data/address lines — BDAL<15:00>
- Two multiplexed address/parity lines — BDAL<17:16>
- Four extended address lines — BDAL<21:18>
- Six data transfer control lines — BBS7, BDIN, BDOUT, BRPLY, BSYNC, BWTBT
- Six system control lines — BHALT, BREF, BEVNT, BINIT, BDCOK, BPOK
- Ten interrupt control and direct memory access control lines — BIAKO, BIAKI, BIRQ4, BIRQ5, BIRQ6, BIRQ7, BDMGO, BDMR, BSACK, BDMGI

In addition, a number of power, ground, and spare lines have been defined for the bus. For a detailed description of these lines, please refer to Table E-1.

The discussion in this chapter applies to the general 22-bit physical address capability. In cases where modules utilize 16- or 18-bit physical address space, this discussion applies to the lines that are utilized by those modules.

Most LSI-11 Bus signals are bidirectional and use terminations for a negated (high) signal level. Devices connect to these lines via high-im-

pedance bus receivers and open collector drivers. *The asserted state is produced when a bus driver asserts the line low.* Although bidirectional lines are electrically bidirectional (any point along the line can be driven or received), certain lines are functionally unidirectional. These lines communicate to or from a bus master (or signal source), but not both. Interrupt acknowledge (BIAK) and direct memory access grant (BDMG) signals are physically unidirectional in a daisy-chain fashion. These signals originate at the processor output signal pins. Each is received on device input pins (BIAKI or BDMGI) and conditionally retransmitted via device output pins (BIAKO or BDMGO). These signals are received from higher-priority devices and are retransmitted to lower-priority devices along the bus, establishing the position-dependent priority scheme.

Master/Slave Relationship

Communication between devices on the bus is asynchronous. A master/slave relationship exists throughout each bus transaction. At any time, there is one device that has control of the bus. This controlling device is termed the bus master. The master device controls the bus when communicating with another device on the bus, termed the slave. The bus master (typically the processor or a DMA device) initiates a bus transaction. The slave device responds by acknowledging the transaction in progress and by receiving data from, or transmitting data to, the bus master. LSI-11 Bus control signals transmitted or received by the bus master or bus slave device must complete the sequence according to bus protocol.

The processor controls bus arbitration, i.e., which device becomes bus master at any given time. A typical example of this relationship is the processor, as master, fetching an instruction from memory, which is always a slave. Another example is a disk, as master, transferring data to memory as slave. Communication on the LSI-11 Bus is interlocked so that for certain control signals issued by the master device, there must be a response from the slave in order to complete the transfer. It is the master/slave signal protocol that makes the LSI-11 Bus asynchronous. The asynchronous operation precludes the need for synchronizing with, and waiting for, clock pulses.

Since bus cycle completion by the bus master requires response from the slave device, each bus master must include a time-out error circuit that will abort the bus cycle if the slave device does not respond to the bus transaction within 10 microseconds. The actual time before a time-out error occurs must be longer than the reply time of the slowest peripheral or memory device on the bus.

See Table E-1 for more detail on signal functions.

Table E-1 Signal Assignments

DATA AND ADDRESS

Nomenclature	Pin Assignment
BDAL0	AU2
BDAL1	AV2
BDAL2	BE2
BDAL3	BF2
BDAL4	BH2
BDAL5	BJ2
BDAL6	BK2
BDAL7	BL2
BDAL8	BM2
BDAL9	BN2
BDAL10	BP2
BDAL11	BR2
BDAL12	BS2
BDAL13	BT2
BDAL14	BU2
BDAL15	BV2
BDAL16	AC1
BDAL17	AD1
BDAL18	BC1
BDAL19	BD1
BDAL20	BE1
BDAL21	BF1

CONTROL

Nomenclature	Pin Assignment
	Data Control
BDOUT	AE2
BRPLY	AF2
BDIN	AH2
BSYNC	AJ2
BWTBT	AK2
BBS7	AP2
	Interrupt Control
BIRQ7	BP1
BIRQ6	AB1
BIRQ5	AA1

BIRQ4	AL2
BIAK0	AN2
BIAKI	AM2
	DMA Control
BDMR	AN1
BSACK	BN1
BDMG0	AS2
BMDGI	AR2
	System Control
BHALT	AP1
BREF	AR1
BEVNT	BR1
BINIT	AT2
BDCOK	BA1
BPOK	BB1

POWER AND GROUND

Nomenclature	**Pin Assignment**
+5B (battery) or +12B(battery)	AS1
+12B	BS1
+5B	AV1
+5	AA2
+5	BA2
+5	BV1
+12	AD2
+12	BD2
−12	AB2
−12	BB2
GND	AC2
GND	AJ1
GND	AM1
GND	AT1
GND	BC2
GND	BJ1
GND	BM1
GND	BT1

SPARES

Nomenclature	Pin Assignment
SSpare1	AE1
SSpare3	AH1
SSpare8	BH1
SSpare2	AF1
MSpareA	AK1
MSpareB	AL1
MSpareB	BK1
MSpareB	BL1
PSpare1	AU1
ASpare2	BU1

DATA TRANSFER BUS CYCLES

Data transfer bus cycles are listed and defined in Table E-2.

Table E-2 Data Transfer Operations

Bus Cycle Mnemonic	Description	Function (with Respect to the Bus Master)
DATI	Data word input	Read
DATO	Data word output	Write
DATOB	Data byte output	Write byte
DATIO	Data word input/output	Read-modify-write
DATIOB	Data word input/ byte output	Read-modify-write byte
DATBI	Data block input	Read block
DATBO	Data block output	Write block

These bus cycles, executed by bus master devices, transfer 16-bit words or 8-bit bytes to or from slave devices. In block mode, multiple words may be transferred to sequential word addresses, starting from a single bus address. The bus signals listed in Table E-3 are used in the data transfer operations described in Table E-2.

Table E-3 Bus Signals for Data Transfers

Mnemonic	Description	Function
BDAL<21:00> L	22 Data/address lines	BDAL<15:00> L are used for word and byte transfers. BDAL<17:16> L are used for extended addressing, memory parity error (16), and memory parity error enable (17) functions. BDAL<21:18> L are used for extended addressing beyond 256 KB.
BSYNC L	Bus Cycle Control	Indicates bus transaction in progress.
BDIN L	Data input indicator	Strobe signals.
BDOUT L	Data output indicator	Strobe signals.
BRPLY L	Slave's acknowledge of bus cycle	Strobe signals.
BWTBT L	Write/byte control	Control signals.
BBS7	I/O device select	Indicates address is in the I/O page.

Data transfer bus cycles can be reduced to five basic types: DATI, DATO(B), DATIO(B), DATBI, and DATBO. These transactions occur between the bus master and one slave device selected during the addressing portion of the bus cycle.

Bus Cycle Protocol

Before initiating a bus cycle, the previous bus transaction must have been completed (BSYNC L negated) and the device must become bus master. The bus cycle can be divided into two parts, an addressing portion, and a data transfer portion. During the addressing portion, the bus master outputs the address for the desired slave device, memory location or device register. The selected slave device responds by latching the address bits and holding this condition for the duration of

the bus cycle until BSYNC L becomes negated. During the data transfer portion, the actual data transfer occurs.

Device Addressing — The device addressing portion of a data transfer bus cycle comprises an address setup and deskew time and an address hold and deskew time. During the address setup and deskew time, the bus master does the following:

- Asserts BDAL<21:00> L with the desired slave device address bits
- Asserts BBS7 L if a device in the I/O page is being addressed
- Asserts BWTBT L if the cycle is a DATO(B) or DATBO bus cycle

During this time the address, BBS7 L, and BWTBT L signals are asserted at the slave bus receiver for at least 75 ns before BSYNC goes active. Devices in the I/O page ignore the nine high-order address bits BDAL<21:13> and instead decode BBS7 L along with the thirteen low-order address bits. An active BWTBT L signal during address setup time indicates that a DATO(B) or DATBO operation follows, while an inactive BWTBT L indicates a DATI, DATBI or DATIO(B) operation.

The address hold and deskew time begins after BSYNC L is asserted.

The slave device uses the active BSYNC L bus receiver output to clock BDAL address bits, BBS7 L, and BWTBT L into its internal logic. BDAL<21:00> L, BBS7 L, and BWTBT L will remain active for 25 ns (minimum) after BSYNC L bus receiver goes active. BSYNC L remains active for the duration of the bus cycle.

Memory and peripheral devices are addressed similarly except for the way the slave device responds to BBS7 L. Addressed peripheral devices must not decode address bits on BDAL<21:13> L. Addressed peripheral devices may respond to a bus cycle when BBS7 L is asserted (low) during the addressing portion of the cycle. When asserted, BBS7 L indicates that the device address resides in the I/O page (the upper 4K address space). Memory devices generally do not respond to addresses in the I/O page; however, some system applications may permit memory to reside in the I/O page for use as DMA buffers, read-only memory bootstraps or diagnostics, etc.

DATI — The DATI bus cycle, illustrated in Figure E-1, is a read operation. During DATI, data are input to the bus master. Data consist of 16-bit word transfers over the bus. During the data transfer portion of the DATI bus cycle, the bus master asserts BDIN L 100 ns minimum after BSYNC L is asserted. The slave device responds to BDIN L active as follows:

- Asserts BRPLY L 0 ns minimum (8 μs maximum to avoid bus timeout) after receiving BDIN L and 125 ns (maximum) before BDAL bus driver data bits are valid.
- Asserts BDAL<21:00> L with the addressed data and error information 0 ns minimum after receiving BDIN and 125 ns maximum after assertion of BRPLY.

When the bus master receives BRPLY L, it does the following:

- Waits at least 200 ns deskew time and then accepts input data at BDAL<17:00> L bus receivers. BDAL<17:16> L are used for transmitting parity errors to the master.
- Negates BDIN L 200 ns (minimum) to 2 microseconds (maximum) after BRPLY L goes active.

The slave device responds to BDIN L negation by negating BRPLY L and removing read data from BDAL bus drivers. BRPLY L must be negated 100 ns (maximum) prior to removal of read data. The bus master responds to the negated BRPLY L by negating BSYNC L.

Conditions for the next BSYNC L assertion are as follows:

- BSYNC L must remain negated for 200 ns (minimum)
- BSYNC L must not become asserted within 300 ns of previous BRPLY L negation

Figure E-2 illustrates DATI bus cycle timing.

NOTE

Continuous assertion of BSYNC L retains control of the bus by the bus master, and the previously addressed slave device remains selected. This is done for DATIO(B) bus cycles where DATO or DATOB follows a DATI without BSYNC L negation and a second device addressing operation. Also, a slow slave device can hold off data transfers to itself by keeping BRPLY L asserted, which will cause the master to keep BSYNC L asserted.

DATO(B) — DATO(B), illustrated in Figure E-3, is a write operation. Data are transferred in 16-bit words (DATO) or 8-bit bytes (DATOB) from the bus master to the slave device. The data transfer output can occur after the addressing portion of a bus cycle when BWTBT L has been asserted by the bus master, or immediately following an input transfer part of a DATIO(B) bus cycle.

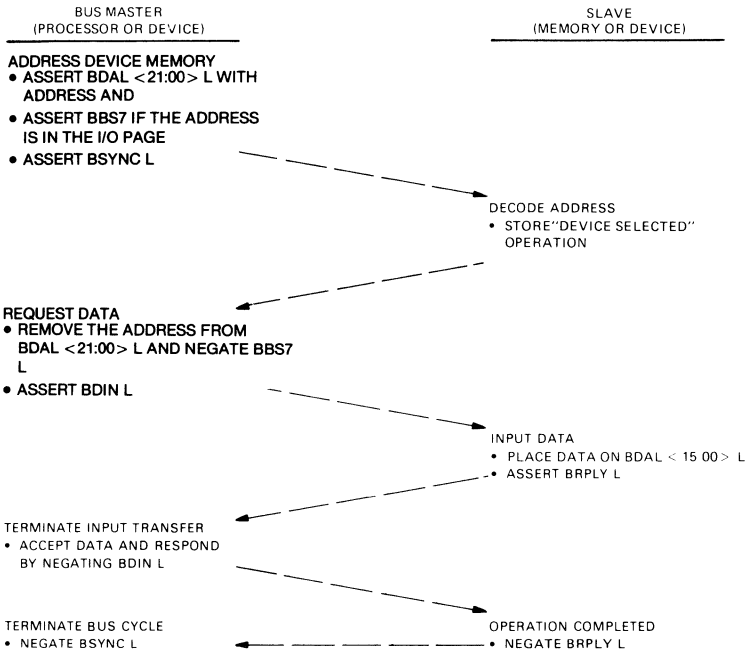
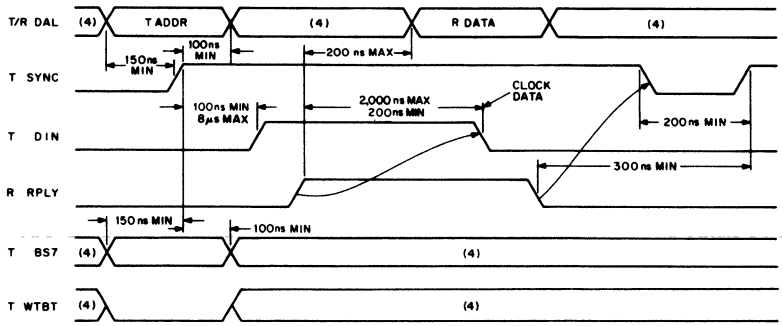


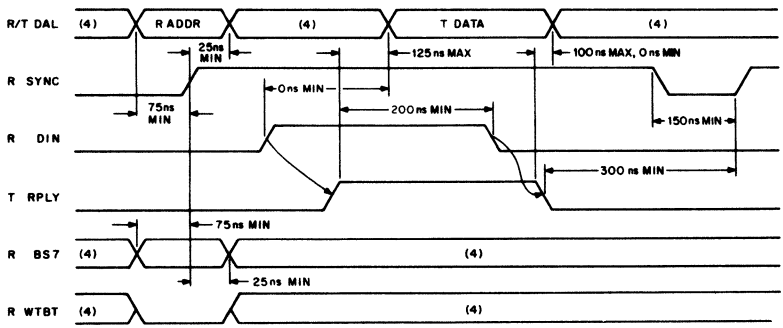
Figure E-1 DATI Bus Cycle

The data transfer portion of a DATO(B) bus cycle comprises a data setup and deskew time and a data hold and deskew time.

During the data setup and deskew time, the bus master outputs the data on BDAL<15:00> L at least 100 ns after BSYNC L is asserted if the transfer is a word transfer. If it is a word transfer, the bus master negates BWTBT L at least 100 ns after BSYNC L assertion. BWTBT L remains negated for the length of the bus cycle. If the transfer is a byte transfer, BWTBT L remains asserted. If it is the output of a DATIOB, BTWBT L becomes asserted and lasts the duration of the bus cycle.



TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES

1 Timing shown at Master and Slave Device Bus Driver inputs and Bus Receiver Outputs

2 Signal name prefixes are defined below
T = Bus Driver Input
R = Bus Receiver Output

3 Bus Driver Output and Bus Receiver Input signal names include a "B" prefix

4 Don't care condition

Figure E-2 DATI Bus Cycle Timing

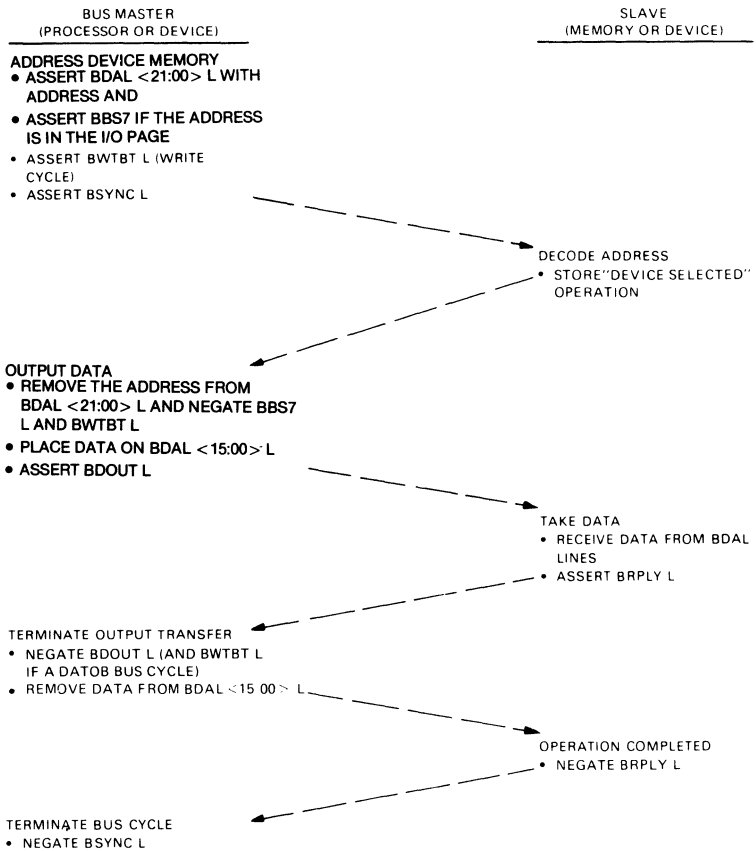
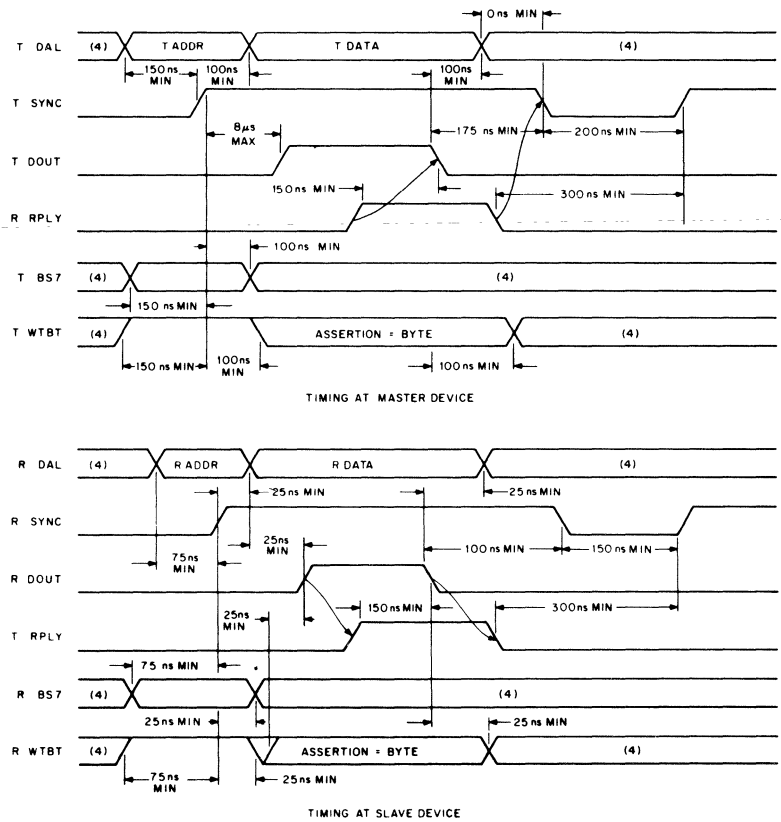


Figure E-3 DATO or DATOB Bus Cycle

During a byte transfer, BDAL <00> L selects the high or low byte. This occurs while in the addressing portion of the cycle. If asserted, the high byte (BDAL<15:08> L) is selected; otherwise, the low byte (BDAL<07:00> L) is selected. An asserted BDAL16 L at this time will force a parity error to be written into memory if the memory is a parity-type memory. BDAL17 L is not used for write operations. The bus master asserts BDOUT L at least 100 ns after BDAL and BWTBT L bus drivers are stable. The slave device responds by asserting BRPLY L within 10 microseconds to avoid bus time-out. This completes the data setup and deskew time.



NOTES

1. Timing shown at Master and Slave Device Bus Driver Inputs and Bus Receiver Outputs
2. Signal name prefixes are defined below
   T = Bus Driver Input
   R = Bus Receiver Output
3. Bus Driver Output and Bus Receiver Input signal names include a "B" prefix
4. Don't care condition

Figure E-4 DATO or DATOB Bus Cycle Timing

During the data hold and deskew time the bus master receives BRPLY L and negates BDOUT L. BDOUT L must remain asserted for at least 150 ns from the receipt of BRPLY L before being negated by the bus master. BDAL<17:00> L bus drivers remain asserted for at least 100 ns after BDOUT L negation. The bus master then negates BDAL inputs.

During this time, the slave device senses BDOUT L negation. The data are accepted and the slave device negates BRPLY L. The bus master responds by negating BSYNC L. However, the processor will not negate BSYNC L for at least 175 ns after negating BDOUT L. This completes the DATO(B) bus cycle. Before the next cycle BSYNC L must remain unasserted for at least 200 ns. Figure E-4 illustrates DATO(B) bus cycle timing.

DATIO(B) — The protocol for a DATIO(B) bus cycle is identical to the addressing and data transfer portions of the DATI and DATO(B) bus cycles, and is illustrated in Figure E-5. After addressing the device, a DATI cycle is performed as explained earlier; however, BSYNC L is not negated. BSYNC L remains active for an output word or byte transfer [DATO(B)]. The bus master maintains at least 200 ns between BRPLY L negation during the DATI cycle and BDOUT L assertion. The cycle is terminated when the bus master negates BSYNC L, as described for DATO(B). Figure E-6 illustrates DATIO(B) bus cycle timing.

DIRECT MEMORY ACCESS

The direct memory access (DMA) capability allows direct data transfer between I/O devices and memory. This is useful when using mass storage devices (e.g., disks) that move large blocks of data to and from memory. A DMA device needs to know only the starting address in memory, the starting address in mass storage, the length of the transfer, and whether the operation is read or write. When this information is available, the DMA device can transfer data directly to or from memory. Since most DMA devices must perform data transfers in rapid succession or lose data, DMA devices are provided the highest priority.

DMA is accomplished after the processor (normally bus master) has passed bus mastership to the highest-priority DMA device that is requesting the bus. The processor arbitrates all requests and grants the bus to the DMA device located electrically closest to it. A DMA device remains bus master indefinitely until it relinquishes its mastership. The following control signals are used during bus arbitration.

BDMGI L	DMA Grant Input
BDMGO L	DMA Grant Output
BDMR L	DMA Request Line
BSACK L	Bus Grant Acknowledge

DMA Protocol

A DMA transaction can be divided into three phases:

- Bus mastership acquisition phase

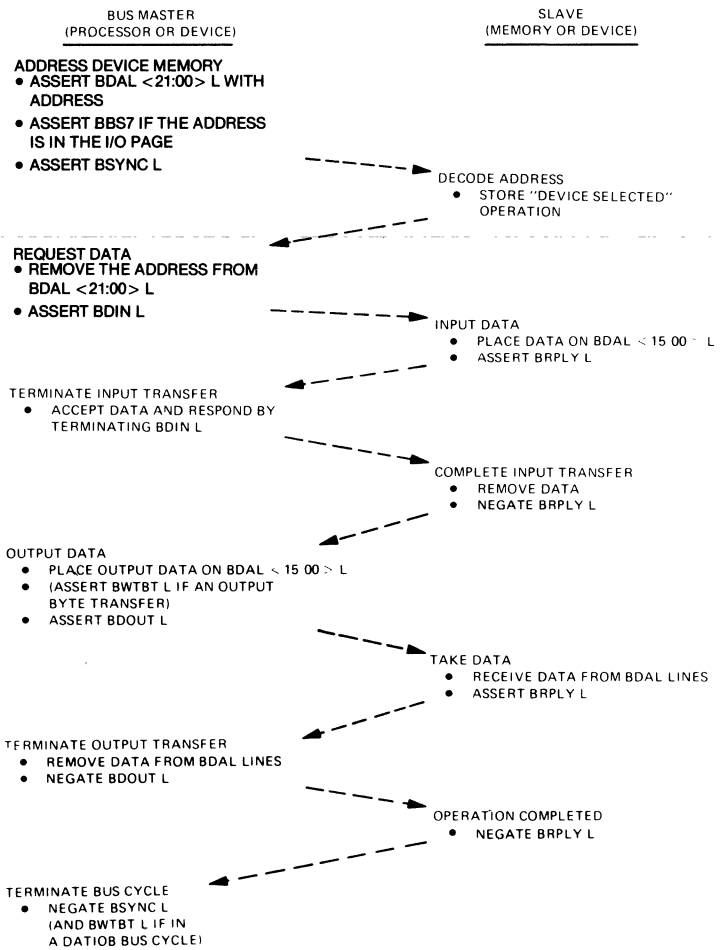
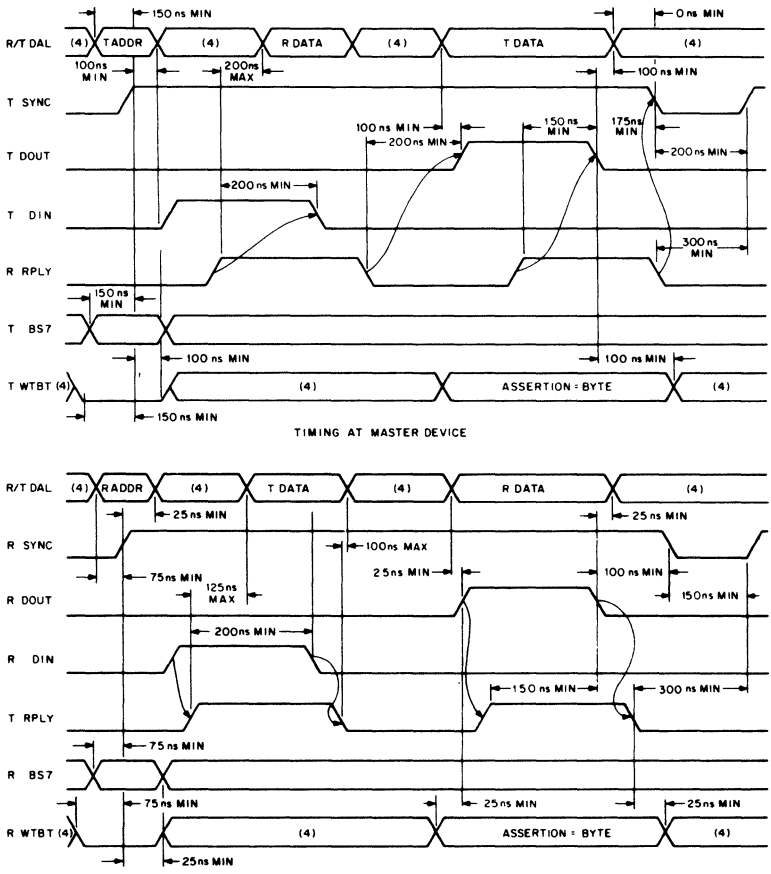


Figure E-5 DATIO or DATIOB Bus Cycle

- Data transfer phase
- Bus mastership relinquish phase

During the bus mastership acquisition phase, a DMA device requests the bus by asserting BDMR L. The processor arbitrates the request and initiates the transfer of bus mastership by asserting BDMGO L.



NOTES

1 Timing shown at Requesting Device
Bus Driver Inputs and Bus Receiver Outputs

2 Signal name prefixes are defined below
T = Bus Driver Input
R = Bus Receiver Output

3 Bus Driver Output and Bus Receiver Input signal names include a "B" prefix

4 Don't care condition

Figure E-6 DATIO or DATIOB Bus Cycle Timing

The maximum time between BDMR L assertion and BDMGO L assertion is DMA latency. This time is processor-dependent. BDMGO L/ BDMGI L is one signal that is daisy-chained through each module in the backplane. It is driven out of the processor on the BDMGO L pin,

enters each module on the BDMGI L pin and exits on the BDMGO L pin. This signal passes through the modules in descending order of priority until it is stopped by the requesting device. The requesting device blocks the output of BMDGO L and asserts BSACK L. If BDMR L is continuously asserted, the bus will be hung.

During the data transfer phase, the DMA device continues asserting BSACK L. The actual data transfer is performed as described earlier.

The DMA device can assert BSYNC L for a data transfer 250 ns (minimum) after it receives BDMGI L and its BSYNC L bus receiver becomes negated.

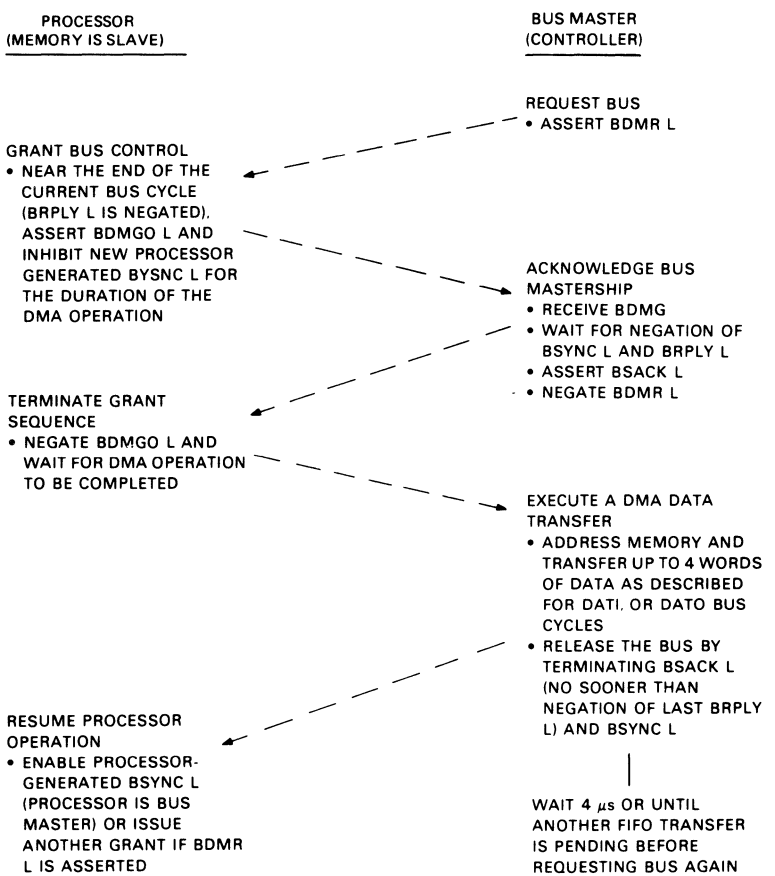


Figure E-7 DMA Protocol

During the bus mastership relinquish phase, the DMA device relinquishes the bus by negating BSACK L. This occurs after completing (or aborting) the last data transfer cycle (BRPLY L negated). BSACK L may be negated up to a maximum of 300 ns before negating BSYNC L. Figure E-7 illustrates the DMA protocol and Figure E-8 illustrates DMA request/grant timing.

NOTE

If multiple data transfers are performed during this phase, consideration must be given to the use of the bus for other system functions, such as memory refresh (if required).

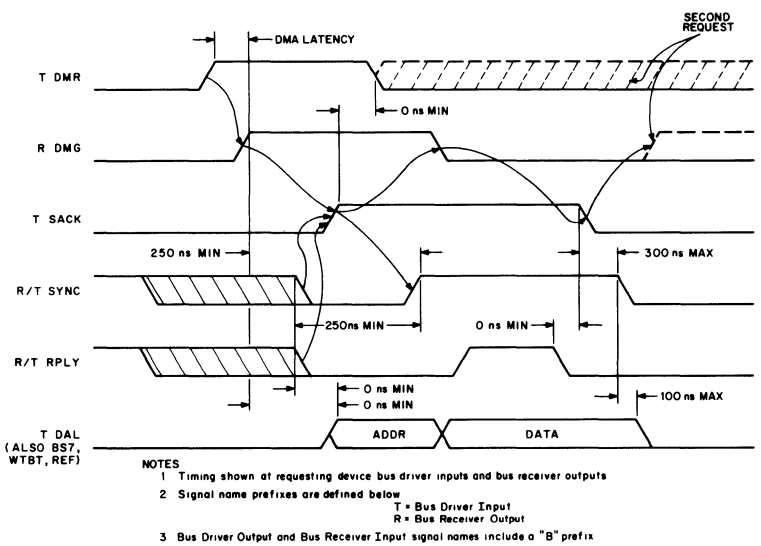


Figure E-8 DMA Request/Grant Timing

BLOCK MODE DMA

For increased throughput, block mode DMA may be implemented on a device for use with memories that support this type of transfer. In a block mode transaction, the starting memory address is asserted, followed by data for that address, and data for consecutive addresses.

By eliminating the assertion of the address for each data word, the transfer rate is almost doubled. The DATBI and DATBO bus cycles are described below.

DATBI

The device addressing portion of the cycle is the same as described earlier for other bus cycles. The bus master gates BDAL <21:00>, BBS7, and the negation of BWTBT onto the bus.

The master asserts the first BDIN 100 ns after BSYNC, and asserts BBS7 a maximum of 50 ns after asserting BDIN for the first time. BBS7 is a request to the slave for a block mode transfer. BBS7 remains asserted until a maximum of 50 ns after the assertion of BDIN for the last time. BBS7 may be gated as soon as the conditions for asserting BDIN are met.

The slave asserts BRPLY a minimum of 0 ns (8 μs maximum to avoid bus timeout) after receiving BDIN. It asserts BREF concurrently with BRPLY if it is a block mode device capable of supporting another BDIN after the current one. The slave gates BDAL <15:00> onto the bus a minimum of 0 ns after the assertion of BDIN and 125 ns maximum after the assertion of BRPLY.

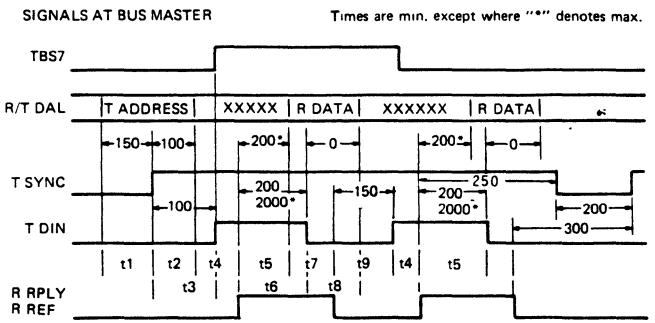
The master receives the stable data from 200 ns maximum after the assertion of BRPLY until 20 ns minimum after the negation of BDIN. It negates BDIN a minimum of 200 ns after the assertion of BRPLY.

The slave negates BRPLY a minimum of 0 ns after the negation of BDIN. If BBS7 and BREF are both asserted when BRPLY is negated, the slave prepares for another BDIN cycle. BBS7 is stable from 125 ns after BDIN is asserted until 150 ns after BRPLY is negated. The master asserts BDIN a minimum of 150 ns after BRPLY is negated and the cycle is continued as before. (BBS7 remains asserted and the slave responds to BDIN with BRPLY and BREF.) BREF is stable from 75 ns after BRPLY is asserted until a minimum of 20 ns after BDIN is negated.

If BBS7 and BREF are not both asserted when BRPLY is negated, the slave removes the data from the bus a minimum of 0 ns and 100 ns maximum after negating BRPLY. The master negates BSYNC a minimum of 250 ns after the assertion of the last BRPLY and a minimum of 0 ns after the negation of that BRPLY.

DATBO

The device addressing portion of the cycle is the same as described earlier. The bus master gates BDAL <21:00>, BBS7, and the assertion of BWTBT onto the bus.



t1	=	address to T SYNC 150ns MIN.	
t2	=	address hold	100ns min
t3	=	T SYNC to T DIN	100ns min
t4	=	T DIN to R RPLY T (drive) + T (prop) + T (receive) + T (delay) + T (drive) + T (prop) + T (receive)	
t5	=	R RPLY to data	200ns max
t6	=	R RPLY to T DIN	200ns min
t7	=	T DIN to R RPLY	
	=	T (drive) + (prop) + T (receive) + T (delay) + T (drive + T (prop) + T (receive)	
t8	=	R RPLY to data	0ns min
t9	=	R RPLY to T DIN	150ns min
T cell	=	t4 + t6 + t7 + t9	—since t6 must be > t5 for master to have valid data - and t9 >t8

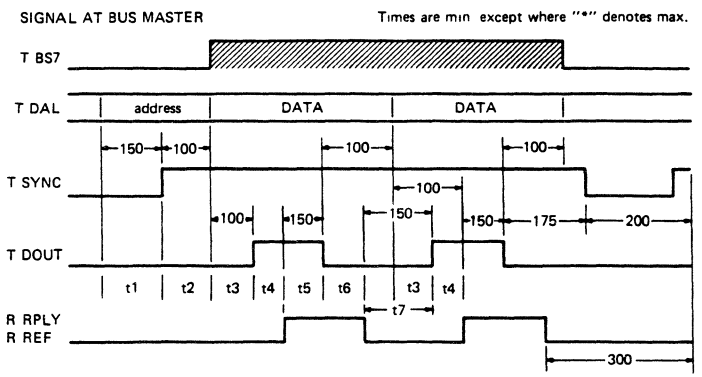
Figure E-9 DATBI Bus Cycle Timing

A minimum of 100 ns after BSYNC is asserted, data on BDAL <15:00> and the negated BWTBT are put onto the bus. The master then asserts BDOUT a minimum of 100 ns after gating the data.

The slave receives stable data and BWTBT from a minimum of 25 ns before the assertion of BDOUT to a minimum of 25 ns after the negation of BDOUT. The slave asserts BRPLY a minimum of 0 ns after receiving BDOUT. It also asserts BREF concurrently with BRPLY if it is a block mode device capable of supporting another BDOUT after the current one.

The master negates BDOUT 150 ns minimum after the assertion of BRPLY. If BREF was asserted when BDOUT was negated and the master wants to transmit more data in this block mode cycle, then the new data is gated onto the bus 100 ns minimum after BDOUT is negated. BREF is stable from 75 ns maximum after BRPLY is asserted until 20 ns minimum after BDOUT is negated. The master asserts BDOUT 100 ns minimum after gating new data onto the bus and 150 ns minimum after BRPLY negates. The cycle continues as before.

If BREF was not asserted when BDOUT was negated or if the bus master does not want to transmit more data in this cycle, then the master removes data from the bus a minimum of 100 ns after negating BDOUT. The slave negates BRPLY a minimum of 0 ns after negating BDOUT. The bus master negates BSYNC a minimum of 175 ns after negating BDOUT, and a minimum of 0 ns after the negation of BRPLY.



t1	=	address to T SYNC	150ns min
t2	=	address hold	100ns min
t3	=	data to T DOUT	100ns min
t4	=	T DOUT to R RPLY	
	=	T (drive) + T (prop) + T (receive) + T (delay) + T (drive) + T (prop) + T (receive)	
t5	=	R RPLY to T DOUT	150ns min
t6	=	T DOUT to R RPLY	
	=	T (drive) + T (prop) + T (receive) + T (delay) + T (drive) + T (prop) + T (receive)	
t7	=	R RPLY to T DOUT	150ns min
T cell	=	t3 + t4 + t5 + t6 + t7	—since t3 < t7

Figure E-10 DATBO Bus Cycle Timing

DMA Guidelines

- Systems with memory refresh over the bus must not include devices that perform more than one transfer per acquisition.
- Bus masters that do not use block mode are limited to four DATI, four DATO, or two DATIO transfers per acquisition.
- Block mode bus masters that do not monitor BDMR are limited to eight transfers per acquisition.
- If BDMR is not asserted after the seventh transfer, block mode bus masters that do monitor BDMR may continue making transfers until the bus slave fails to assert BREF or until they reach the total maximum of 16 transfers. Otherwise, they stop after eight transfers.

INTERRUPTS

The interrupt capability of the LSI-11 Bus allows any I/O device to temporarily suspend (interrupt) current program execution and divert processor operation to service the requesting device. The processor inputs a vector from the device to start the service routine (handler). Like the device register address, hardware fixes the device vector at locations within a designated range below location 001000. The vector indicates the first of a pair of addresses. The content of the first address is read by the processor and is the starting address of the interrupt handler. The content of the second address is a new processor status word (PS). The new PS can raise the interrupt priority level, thereby preventing lower- level interrupts from breaking into the current interrupt service routine. Control is returned to the interrupted program when the interrupt handler is ended. The original interrupted program's address (PC) and its associated PS are stored on a stack. The original PC and PS are restored by a return from interrupt (RTI or RTT) instruction at the end of the handler. The use of the stack and the LSI-11 Bus interrupt scheme can allow interrupts to occur within interrupts (nested interrupts), depending on the PS.

Interrupts can be caused by LSI-11 Bus options or the CPU. Those interrupts that originate from within the processor are called traps. Traps are caused by programming errors, hardware errors, special instructions, and maintenance features.

The LSI-11 Bus signals used in interrupt transactions are:

BIRQ4 L	Interrupt request priority level 4
BIRQ5 L	Interrupt request priority level 5
BIRQ6 L	Interrupt request priority level 6
BIRQ7 L	Interrupt request priority level 7

BIAKI L	Interrupt acknowledge input
BIAKO L	Interrupt acknowledge output
BDAL<21:00> L	Data/address lines
BDIN L	Data input strobe
BRPLY L	Reply

Device Priority

The LSI-11 Bus supports the following two methods of device priority:

- Distributed Arbitration—priority levels are implemented on the hardware. When devices of equal priority level request an interrupt, priority is given to the device electrically closest to the processor.
- Position-Defined Arbitration—priority is determined solely by electrical position on the bus. The closer a device is to the processor, the higher its priority is.

Interrupt Protocol

Interrupt protocol on the LSI-11/23 has three phases: interrupt request phase, interrupt acknowledge, and priority arbitration phase, and interrupt vector transfer phase. Figure E-11 illustrates the interrupt request/acknowledge sequence.

The interrupt request phase begins when a device meets its specific conditions for interrupt requests. For example, the device is ready, done, or an error has occurred. The interrupt enable bit in a device status register must be set. The device then initiates the interrupt by asserting the interrupt request line(s). BIRQ4 L is the lowest hardware priority level and is asserted for all interrupt requests for compatibility with previous LSI-11 processors. The level a device is configured at must also be asserted. A special case exists for level 7 devices which must also assert level 6. See the arbitration discussion below involving the 4-level scheme for an explanation.

Interrupt Level	Lines Asserted by Device
4	BIRQ4 L
5	BIRQ4 L, BIRQ5 L
6	BIRQ4 L, BIRQ6 L
7	BIRQ4 L, BIRQ6 L, BIRQ7 L

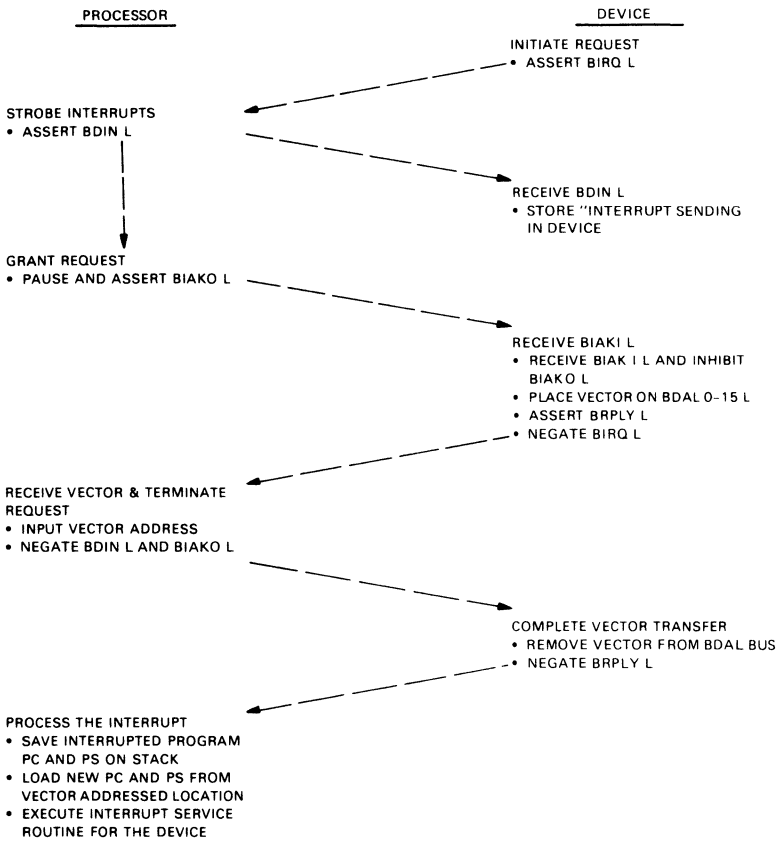


Figure E-11 Interrupt Request/Acknowledge Sequence

The interrupt request line remains asserted until the request is acknowledged.

During the interrupt acknowledge and priority arbitration phase the LSI-11/23 processor will acknowledge interrupts under the following conditions:

1. The device interrupt priority is higher than the current PS<7:5>.
2. The processor has completed instruction execution and no additional bus cycles are pending.

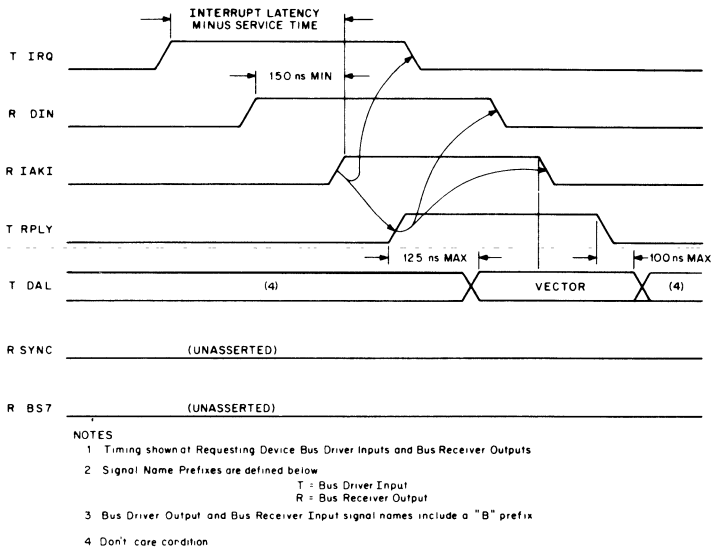


Figure E-12 Interrupt Protocol Timing

The processor acknowledges the interrupt request by asserting BDIN L, and 150 ns (minimum) later asserting BIAKO L. The device electrically closest to the processor receives the acknowledge on its BIAKI L bus receiver.

At this point the two types of arbitration must be discussed separately. If the device that receives the acknowledge uses the 4-level interrupt scheme, it reacts as described below:

1. If not requesting an interrupt, the device asserts BIAKO L and the acknowledge propagates to the next device on the bus.
2. If the device is requesting an interrupt, it must check to see that no higher-level device is currently requesting an interrupt. This is done by monitoring higher-level request lines. The table below lists the lines that need to be monitored by devices at each priority level.

   In addition to asserting levels 7 and 4, level 7 devices must drive level 6. This is done to simplify the monitoring and arbitration by level 4 and 5 devices. In this protocol, level 4 and 5 devices need not monitor level 7 since level 7 devices assert level 6. Level 4 and 5 devices will become aware of a level 7 request since they moni-

tor the level 6 request. This protocol has been optimized for level 4, 5, and 6 devices, since level 7 devices very seldom are necessary.

Device Priority Level	Line(s) Monitored
4	BIRQ5, BIRQ6
5	BIRQ6
6	BIRQ7
7	—

3. If no higher-level device is requesting an interrupt, the acknowledge is blocked by the device. (BIAKO L is not asserted.) Arbitration logic within the device uses the leading edge of BDIN L to clock a flip-flop that blocks BIAKO L. Arbitration is won, and the interrupt vector transfer phase begins.
4. If a higher-level request line is active, the device disqualifies itself and asserts BIAKO L to propagate the acknowledge to the next device along the bus.

Signal timing must be carefully considered when implementing 4-level interrupts. Note Figure E-12.

If a single-level interrupt device receives the acknowledge, it reacts as follows:

- If not requesting an interrupt, the device asserts BIAKO L and the acknowledge propagates to the next device on the bus.
- If the device was requesting an interrupt, the acknowledge is blocked using the leading edge of BDIN L and arbitration is won. The interrupt vector transfer phase begins.

The interrupt vector transfer phase is enabled by BDIN L and BIAKI L. The device responds by asserting BRPLY L and its BDAL<15:00> L bus driver inputs with the vector address bits. The BDAL bus driver inputs must be stable within 125 ns (maximum) after BRPLY L is asserted. The processor then inputs the vector address and negates BDIN L and BIAKO L. The device then negates BRPLY L and 100 ns (maximum) later removes the vector address bits. The processor then enters the device's service routine.

NOTE

Propagation delay from BIAKI L to BIAKO L must not be greater than 500 ns per LSI-11 Bus slot.

The device must assert BRPLY L within 10 microseconds (maximum) after the processor asserts BIAKI L.

LSI-11/23 Four-Level Interrupt Configurations

If you have high-speed peripherals and desire better software performance, you can use the 4-level interrupt scheme. Both position-independent and position-dependent configurations can be used with the 4-level interrupt scheme.

The position-independent configuration is illustrated in Figure E-13. This allows peripheral devices that use the 4-level interrupt scheme to be placed in the backplane in any order. These devices must send out interrupt requests and monitor higher-level request lines as described. The level 4 request is always asserted by a requesting device regardless of priority, to allow compatibility if an LSI-11 or LSI-11/2 processor is in the same system. If two or more devices of equally high priority request an interrupt, the device physically closest to the processor will win arbitration. Devices that use the single-level interrupt scheme must be modified or placed at the end of the bus for arbitration to function properly.

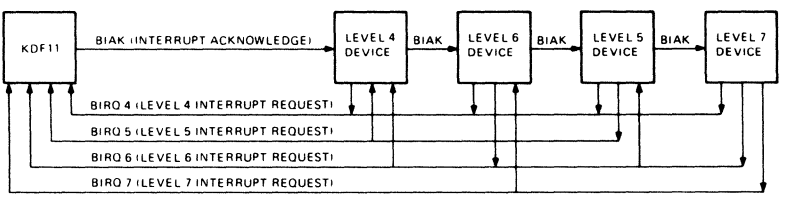


Figure E-13 Position-Independent Configuration

The position-dependent configuration is illustrated in Figure E-14. This configuration is simpler to implement. A constraint is that peripheral devices must be inserted with the highest-priority device located closest to the processor and the remaining devices placed in the backplane in decreasing order of priority, with the lowest-priority devices farthest from the processor. With this configuration each device has to assert only its own level and level 4 (for compatibility with an LSI-11 or LSI-11/2). Monitoring higher level request lines is unnecessary. Arbitration is achieved through the physical positioning of each device on the bus. Single-level interrupt devices on level 4 should be positioned last on the bus.

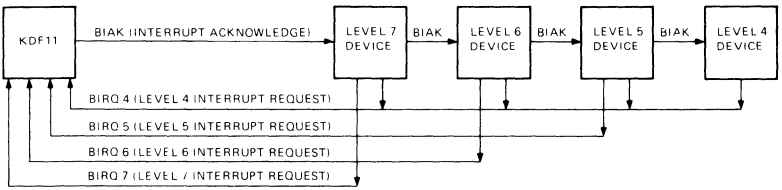


Figure E-14 Position-Dependent Configuration

CONTROL FUNCTIONS

The following LSI-11 Bus signals provide control functions.

BREF L	Memory refresh (also block mode DMA)	
	BHALT L	Processor halt
	BINIT L	Initialize
	BPOK H	Power OK
	BDCOK H	DC power OK

Memory Refresh

If BREF is asserted during the address portion of a bus data transfer cycle, it causes all dynamic MOS memories to be addressed simultaneously. The sequence of addresses required for refreshing the memories is determined by the specific requirements for each memory. The complete memory refresh cycle consists of a series of refresh bus transactions. A new address is used for each transaction. A complete memory refresh cycle must be completed within 1 or 2 ms. Multiple data transfers by DMA devices must be avoided since they could delay memory refresh cycles. This type of refresh is done only for memories which do not perform on-board refresh.

Halt

Assertion of BHALT L for at least 25 μs interrupts the processor, which stops program execution and forces the processor unconditionally into console ODT mode.

Initialization

Devices along the bus are initialized when BINIT L is asserted. The processor can assert BINIT L as a result of executing a RESET instruction as part of a power-up or power-down sequence, or after detection of a G character in ODT. BINIT L is asserted for approximately 10 microseconds when RESET is executed.

Power Status

Power status protocol is controlled by two signals, BPOK H and BDCOK H. These signals are driven by some external device (usually the power supply).

BDCOK H — When asserted, this indicates that dc power has been stable for at least 3 ms. Once asserted, this line remains asserted until the power fails. It indicates that only 5 microseconds of dc power reserve remains.

BPOK H — When asserted, this indicates that there is at least an 8 ms reserve of dc power and that BDCOK H has been asserted for at least 70 ms. Once BPOK H has been asserted, it must remain asserted for at least 3 ms. The negation of this line, the first event in the power-fail sequence, indicates that power is failing and that only 4 ms of dc power reserve remains.

Power-Up/Down Protocol

Power-up protocol begins when the power supply applies power with BDCOK H negated. This forces the processor to assert BINIT L. When the dc voltages are stable, the power supply or other external device asserts BDCOK H. The processor responds by clearing the PS, floating point status register (FPS), and floating point exception register (FEC). BINIT L is asserted for 12.6 microseconds and then negated for 110 microseconds. The processor continues to test for BPOK H until it is asserted. The power supply asserts BPOK H 70 ms (minimum) after BDCOK H is asserted. The processor then performs its power-up sequence. Normal power must be maintained at least 3.0 ms before a power-down sequence can begin.

A power-down sequence begins when the power supply negates BPOK H. When the current instruction is completed, the processor traps to a power-down routine at location 24_8. The end of the routine is terminated with a HALT instruction to avoid any possible memory corruption as the dc voltages decay.

When the processor executes the HALT instruction, it tests the BPOK H signal. If BPOK H is negated, the processor enters the power-up sequence. It clears internal registers, generates BINIT L, and continues to check for the assertion of BPOK H. If it is asserted and dc voltages are still stable, the processor will perform the rest of the power-up sequence. Figure E-15 illustrates power-up/P ower-down timing.

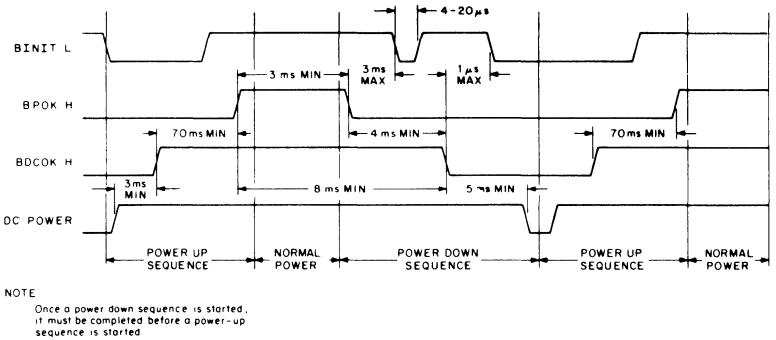


Figure E-15 Power-Up/Power-Down Timing

LSI-11 BUS ELECTRICAL CHARACTERISTICS

Signal Level Specification

Input Logic Levels

TTL Logical Low:	0.8 Vdc maximum
TTL Logical High:	2.0 Vdc minimum

Output Logic Levels

TTL Logical Low:	0.4 Vdc maximum
TTL Logical High:	2.4 Vdc minimum

Load Definition

AC loads comprise the maximum capacitance allowed per signal line to ground. A unit load is defined as 9.35 pF of capacitance. DC loads are defined as maximum current allowed with a signal line driver asserted or unasserted. A unit load is defined as 210 μA in the unasserted state.

120 Ohm LSI-11 Bus

The electrical conductors interconnecting the bus device slots are treated as transmission lines. A uniform transmission line, terminated in its characteristic impedance, will propagate an electrical signal without reflections. Since bus drivers, receivers, and wiring connected to the bus have finite resistance and nonzero reactance, the transmission line impedance is not uniform, and introduces distortions into pulses propagated along it. Passive components of the LSI-11 Bus

(such as wiring, cabling, and etched signal conductors) are designed to have a nominal characteristic impedance of 120 ohms.

The maximum length of interconnecting cable excluding wiring within the backplane is limited to 4.88 m (16 ft.).

Bus Drivers

Devices driving the 120 ohm LSI-11 Bus must have open collector outputs and meet the following specifications.

DC Specifications	Output low voltage when sinking 70 mA of current: 0.7V maximum.
	Output high leakage current when connected to 3.8 Vdc: 25 μA (even if no power is applied, except for BDCOK H and BPOK H).
	These conditions must be met at worst-case supply voltage, temperature, and input signal levels.
AC Specifications	Bus driver output pin capacitive load: Not to exceed 10 pF.
	Propagation delay: Not to exceed 35 ns.
	Skew (difference in propagation time between slowest and fastest gate): Not to exceed 25 ns.
	Rise/Fall Times: Transition time (from 10% to 90% for positive transition, and from 90% to 10% for negative transition) must be no faster than 10 ns.

Bus Receivers

Devices that receive signals from the 120 ohm LSI-11 Bus must meet the following requirements.

DC Specifications	Input low voltage (maximum): 1.3V.
	Input high voltage (minimum): 1.7V.
	Maximum input current when connected to 3.8 Vdc: 80 μA even if no power is applied.
	These specifications must be met at worst-case supply voltage, temperature, and output signal conditions
AC Specifications	Bus receiver input pin capacitance load: Not to exceed 10 pF.

Propagation delay: Not to exceed 35 ns.

Skew (difference in propagation time between slowest and fastest gate): Not to exceed 25 ns.

Bus Termination

The 120 ohm LSI-11 Bus must be terminated at each end by an appropriate terminator, as illustrated in Figure E-16. This is to be done as a voltage divider with its Thevenin equivalent equal to 120 ohms and 3.4V nominal. This type of termination is provided by an REV11-A refresh/boot/terminator, BDV11-AA, KPV11-B, TEV11, or by certain backplanes and expansion cards.

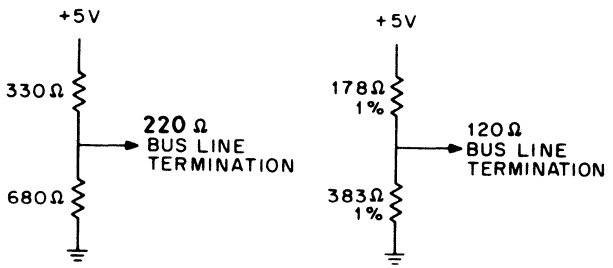


Figure E-16 Bus Line Terminations

Each of the several LSI-11 Bus lines (all signals whose mnemonics start with the letter B) must see an equivalent network with the following characteristics at each end of the bus:

Input impedance (with respect to ground)	120 ohm +5%, −15%
Open circuit voltage	3.4 Vdc +5%
Capacitance Load	Not to exceed 30 pF

NOTE

The resistive termination may be provided by the combination of two modules (i.e., the processor module supplies 220 ohms to ground. This, in parallel with another 220 ohm card provides 120 ohms.) Both of these terminators must be physically resident within the same backplane.

Bus Interconnecting Wiring

Backplane Wiring — The wiring that connects all device interface slots on the LSI-11 must meet the following specifications:

1. The conductors must be arranged such that each line exhibits a characteristic impedance of 120 ohms (measured with respect to the bus common return).
2. Crosstalk between any two lines must be no greater than 5%. Note that worst-case crosstalk is manifested by simultaneously driving all but one signal line and measuring the effect on the undriven line.
3. DC resistance of the signal path, as measured between the near-end terminator and the far-end terminator module (including all intervening connectors, cables, backplane wiring, connector-module etch, etc.) must not exceed 2 ohms.
4. DC resistance of common return path, as measured between the near-end terminator and the far-end terminator module (including all intervening connectors, cables, backplane wiring, connector-module etch, etc.) must not exceed an equivalent of 2 ohms per signal path. Thus, the composite signal return path dc resistance must not exceed 2 ohms divided by 40 bus lines, or 50 milliohms. Note that although this common return path is nominally at ground potential, the conductance must be part of the bus wiring. The specified low impedance return path must be provided by the bus wiring as distinguished from the common system or power ground path.

Intra-Backplane Bus Wiring — The wiring that connects the bus connector slots within one contiguous backplane is part of the overall bus transmission line. Owing to implementation constraints, the nominal characteristic impedance of 120 ohms may not be achievable. Distributed wiring capacitance in excess of the amount required to achieve the nominal 120 ohm impedance may not exceed 60 pF per signal line per backplane.

Power and Ground — Each bus interface slot has connector pins assigned for the following dc voltages. The maximum allowable current per pin is 1.5 A. +5 Vdc must be regulated to ±5% with a maximum ripple of 100 mV pp. +12 Vdc must be regulated to ±3% with a maximum ripple of 200 mV pp.

- +5Vdc—Three pins (4.5 A maximum per bus device slot)
- +12 Vdc—Two pins (3.0 A maximum per bus device slot)
- Ground—Eight pins (shared by power return and signal return)

NOTE

Power is not bused between backplanes on any interconnecting bus cables.

SYSTEM CONFIGURATIONS

LSI-11 Bus systems can be divided into two types:

1. Systems containing one backplane
2. Systems containing multiple backplanes

Before configuring any system, three characteristics for each module in the system must be known. These characteristics are:

- Power consumption— + 5 Vdc and + 12 Vdc current requirements.
- AC bus loading—the amount of capacitance a module presents to a bus signal line. AC loading is expressed in terms of ac loads where one ac load equals 9.35 pF of capacitance.
- DC bus loading—the amount of dc leakage current a module presents to a bus signal when the line is high (undriven). DC loading is expressed in terms of dc loads where one dc load equals 210 microamperes (nominal).

Power consumption, ac loading, and dc loading specifications for each module are included in the *Microcomputer Interface Handbook*.

NOTE

The ac and dc loads and the power consumption of the processor module, terminator module, and backplane must be included in determining the total loading of a backplane.

Rules for Configuring Single Backplane Systems

- When using a processor with 220 ohm termination, the bus can accomodate modules that have up to 20 ac loads (total) before additional termination is required. If more than 20 ac loads are included, the other end of the bus must be terminated with 120 ohms, and then up to 35 ac loads may be present.
- With 120 ohm processor termination, up to 35 ac loads can be used without additional termination. If 120 ohm bus termination is added, up to 45 ac loads can be configured in the backplane.
- The bus can accommodate modules up to 20 dc loads (total).
- The bus signal lines on the backplane can be up to 35.6 cm (14 in.) long.

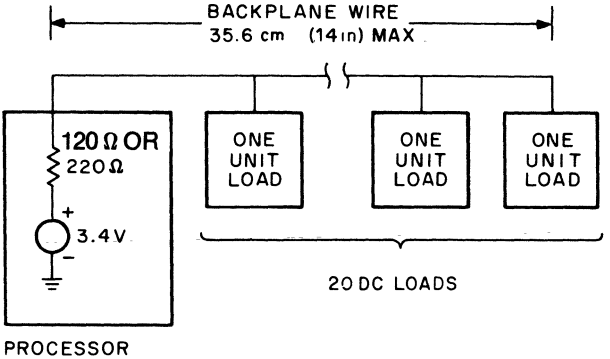


Figure E-17 Single Backplane Configuration

Rules for Configuring Multiple Backplane Systems

- As illustrated in Figure E-18, up to three backplanes may make up the system.
- The signal lines on each backplane can be up to 25.4 cm (10 in.) long.
- Each backplane can accommodate modules that have up to 22 ac loads (total). Unused ac loads from one backplane may not be added to another backplane if the second backplane loading will exceed 22 ac loads. It is desirable to load backplanes equally, or with the highest ac loads in the first and second backplanes.
- DC loading of all modules in all backplanes cannot exceed 20 loads (total).
- Both ends of the bus must be terminated with 120 ohms. This means that the first and last backplane must have an impedance of 120 ohms. To achieve this, each backplane may be lumped together as a single point. The resistive termination may be provided by a combination of two modules in the backplane—the processor providing 220 ohms to ground in parallel with an expansion paddle card providing 250 ohms to give the needed 120 ohm termination. Alternately, a processor with 120 ohm termination would need no additional termination on the paddle card to attain 120 ohms in the first box. The 120 ohm termination in the last box can be provided in two ways. The termination resistors may reside either on the expansion paddle card or on a bus termination card such as the BDV11.

- The cable(s) connecting the first two backplanes are 61 cm (2 ft.) or greater in length.
- The cable(s) connecting the second backplane to the third backplane are 122 cm (4 ft.) longer or shorter than the cable(s) connecting the first and second backplanes.

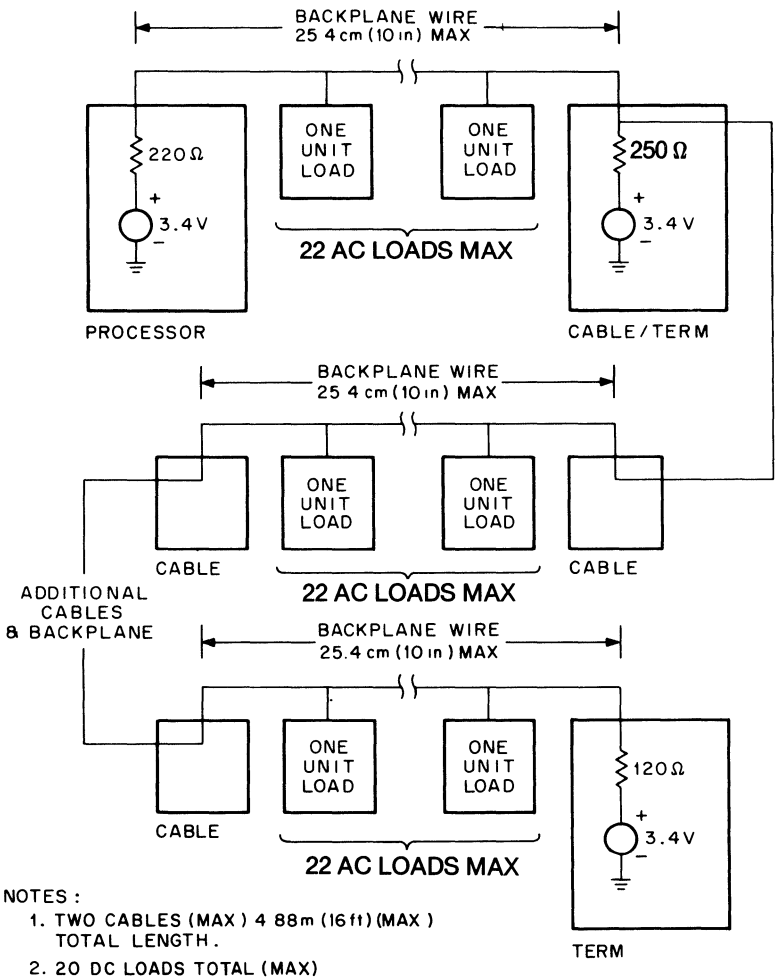


Figure E-18 Multiple Backplane Configuration

- The combined length of both cables cannot exceed 4.88 m (16 ft.).
- The cables used must have a characteristic impedance of 120 ohms.

Power Supply Loading

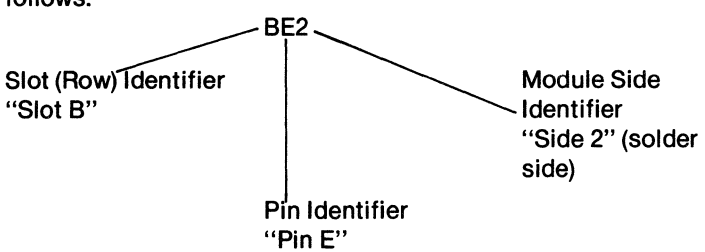
Total power requirements for each backplane can be determined by obtaining the total power requirements for each module in the backplane. Obtain separate totals for +5V and +12V power. Power requirements for each module are specified in the *Microcomputer Interfaces Handbook*.

When distributing power in multiple backplane systems, do not attempt to distribute power via the LSI-11 Bus cables. Provide separate, appropriate power wiring from each power supply to each backplane. Each power supply should be capable of asserting BPOK H and BDCOK H signals according to bus protocol; this is required if automatic power-fail/restart programs are implemented, or if specific peripherals require an orderly power-down halt sequence. The proper use of BPOK H and BDCOK H signals is strongly recommended.

MODULE CONTACT FINGER IDENTIFICATION

DIGITAL plug-in modules all use the same contact finger (pin) identification system. The LSI-11 Bus is based on the use of double-height modules that plug into a 2-slot bus connector. Each slot contains 36 lines (18 each on component and solder sides of circuit board).

Slots, shown as row A and row B in Figure E-19, include a numeric identifier for the side of the module. The component side is designated side 1 and the solder side is designated side 2. Letters ranging from A through V (excluding G, I, O, and Q) identify a particular pin on a side of a slot. Table E-4 lists and identifies the bus pins of the double-height module. The bus pin identifier ending with a 1 is found on the component side of the board, while a bus pin identifier ending with a 2 is found on the solder side of the board. A typical pin is designated as follows.



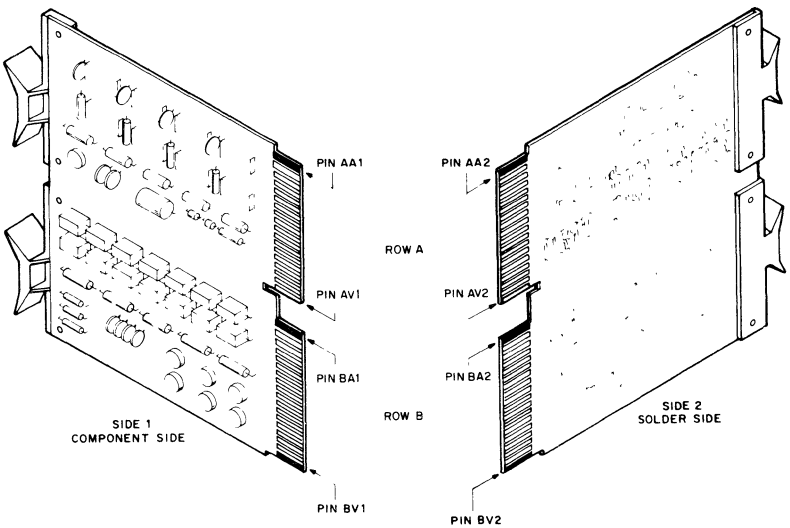


Figure E-19 Double-Height Module Contact Finger Identification

The positioning notch between the two rows of pins mates with a protrusion on the connector block for correct module positioning.

Table E-4 Bus Pin Identifiers

BUS PIN	MNEMONICS	DESCRIPTION
AA1	BIRQ5 L	Interrupt Request Priority Level 5
AB1	BIRQ6 L	Interrupt Request Priority Level 6
AC1	BDAL16 L	Extended address bit during addressing protocol; memory error data line during data transfer protocol.
AD1	BDAL17 L	Extended address bit during addressing protocol; memory error logic enable during data transfer protocol.

BUS PIN	MNEMONICS	DESCRIPTION
AE1	SSPARE1 (Alternate +5B)	Special Spare—not assigned or bused in DIGITAL cable or backplane assemblies; available for user connection. Optionally, this pin may be used for +5V battery (+5B) backup power to keep critical circuits alive during power failures. A jumper is required on LSI-11 Bus options to open (disconnect) the +5B circuit in systems that use this line as SSPARE1.
AF1	SSPARE2	Special Spare—not assigned or bused in DIGITAL cable or backplane assemblies; available for user interconnection. In the highest-priority device slot, the processor may use this pin for a signal to indicate its RUN state.
AH1	SSPARE 3 SRUN simultaneously	Special Spare—not assigned or bused in DIGITAL cable or backplane assemblies; available for user interconnection. An alternate SRUN signal may be connected in the highest-priority set.
AJ1	GND	Ground—System signal ground and dc return.
AK1	MSPAREA	Maintenance Spare—Normally connected together on the backplane at each option location (not bused connection).
AL1	MSPAREB	Maintenance Spare—Normally connected together on the backplane at each option location (not bused connection).

BUS PIN	MNEMONICS	DESCRIPTION
AM1	GND	Ground—System signal ground and dc return.
AN1	BDMR L	Direct Memory Access (DMA) Request—A device asserts this signal to request bus mastership. The processor arbitrates bus mastership between itself and all DMA devices on the bus. If the processor is not bus master (it has completed a bus cycle and BSYNC L is not being asserted by the processor), it grants bus mastership to the requesting device by asserting BDMGO L. The device responds by negating BDMR L and asserting BSACK L.
AP1	BHALT L	Processor Halt—When BHALT L is asserted for at least 25 μs, the processor services the halt interrupt and responds by halting normal program execution. External interrupts are ignored but memory refresh interrupts in LSI-11 are enabled if W4 on M7264 and M7264-YA processor modules is removed and DMA request/grant sequences are enabled. The processor executes the ODT microcode and the console device operation is invoked.
AR1	BREF L	Memory Refresh—Asserted by a DMA device. This signal forces all dynamic MOS memory units requiring bus refresh signals to be activated for each BSYNC L/BDIN L bus transaction. It is also used as a control signal for block mode DMA.

BUS PIN	MNEMONICS	DESCRIPTION
		CAUTION The user must avoid multiple DMA data transfers (burst or "hog" mode) that could delay refresh operation if using DMA refresh. Complete refresh cycles must occur once every 1.6 msec if required.
AS1	+12B or + 5B	+12 Vdc or +5V battery backup power to keep critical circuits alive during power failures. This signal is not bused to BS1 in all DIGITAL backplanes. A jumper is required on all LSI-11 Bus options to open (disconnect) the backup circuit from the bus in systems that use this line at the alternate voltage.
AT1	GND	Ground—System signal ground and dc return.
AU1	PSPARE 1	Spare (Not assigned. Customer usage not recommended.) Prevents damage when modules are inserted upside down.
AV1	+5B	+5V Battery Power—Secondary +5V power connection. Battery power can be used with certain devices.
BA1	BDCOK H	DC Power OK—Power supply-generated signal that is asserted when there is sufficient dc voltage available to sustain reliable system operation.
BB1	BPOK H	Power OK—Asserted by the power supply 70 ms after BDCOK negated when ac pow-

BUS PIN	MNEMONICS	DESCRIPTION
		er drop below the value required to sustain power (approximately 75% of nominal). When negated during processor operation, a power fail trap sequence is initiated.
BC1	SSPARE4 BDAL 18L (22-bit only)	Special Spare in the LSI-11 Bus—Not assigned. Bussed in 22-bit cable and backplane assemblies; available for use interconnection.
BD1	SSPARE5 BDAL 19L (22-bit only)	Caution. These pins may be used as test points by DIGITAL in some options.
BE1	SSPARE6 BDAL 20L	In the 22-bit LSI-11 Bus, these bussed address lines are Address Lines <21:18> currently not used during data time.
BF1	SSPARE7 BDAL 21L	In the 22-bit LSI-11 Bus these bussed address lines are Address Lines <21:18> currently not used during data time.
BH1	SSPARE8	Special Spare—Not assigned or bused in DIGITAL cable and backplane assemblies; available for user interconnection.
BJ1	GND	Ground—System signal ground and dc return.
BK1 BL1	MSPAREB MSPAREB	Maintenance Spare—Normally connected together on the backplane at each option location (not a bused connection).
BM1	GND	Ground—System signal ground and dc return.

BUS PIN	MNEMONICS	DESCRIPTION
BN1	BSACK L	This signal is asserted by a DMA device in response to the processor's BDMGO L signal, indicating that the DMA device is bus master.
BP1	BIRQ7 L	Interrupt request priority level 7
BR1	BEVNT L	External Event Interrupt Request—When asserted, the processor responds by entering a service routine via vector address 100_8. A typical use of this signal is a line time clock interrupt.
BS1	+12B	+12 Vdc battery backup power (not bused to AS1 in all DIGITAL backplanes).
BT1	GND	Ground—System signal ground and dc return.
BU1	PSPARE2	Power Spare 2 (not assigned a function, not recommended for use). If a module is using −12V (on pin AB2) and if the module is accidentally inserted upside down in the backplane, −12 Vdc appears on pin BU1.
BV1	+5	+5V Power—Normal +5 Vdc system power.
AA2	+5	+5V Power—Normal +5 Vdc system power.
AB2	−12	−12V Power—−12 Vdc (optional) power for devices requiring this voltage.
		NOTE LSI-11 modules which require negative voltages contain an inverter circuit (on each module) which generates the re-

BUS PIN	MNEMONICS	DESCRIPTION
		quired voltage(s). Hence, – 12V power is not required with DIGITAL-supplied options.
AC2	GND	Ground—System signal ground and dc return.
AD2	+ 12	+ 12V Power—12 Vdc system power.
AE2	BDOUT L	Data Output—BDOUT, when asserted, implies that valid data is available on BDAL <0:15> L and that an output transfer, with respect to the bus master device, is taking place. BDOUT L is deskewed with respect to data on the bus. The slave device responding to the BDOUT L signal must assert BRPLY L to complete the transfer.
AF2	BRPLY L	Reply—BRPLY L is asserted in response to BDIN L or BDOUT L and during IAK transactions. It is generated by a slave device to indicate that it has placed its data on the BDAL bus or that it has accepted output data from the bus.
AH2	BDIN L	Data Input—BDIN L is used for two types of bus operation: When asserted during BSYNC L time, BDIN L implies an input transfer with respect to the current bus master, and requires a response (BRPLY L). BDIN L is asserted when the master device is ready to accept data from a slave device. When asserted without BSYNC

BUS PIN	MNEMONICS	DESCRIPTION
		L, it indicates that an interrrupt operation is occurring. The master device must deskew input data from BRPLY L.
AJ2	BSYNC L	Synchronize—BSYNC L is asserted by the bus master device to indicate that it has placed an address on BDAL<0:17> L. The transfer is in process until BSYNC L is negated.
AK2	BWTBT L	Write/Byte—BWTBT L is used in two ways to control a bus cycle: It is asserted at the leading edge of BSYNC L to indicate that an output sequence is to follow (DATO or DATOB), rather than an input sequence. It is asserted during BDOUT L, in a DATOB bus cycle, for byte addressing.
AL2	BIRQ4 L	Interrupt Request Priority Level 4— A level 4 device asserts this signal when its interrupt enable and interrupt request flips-flops are set. If the PS word bit 7 is 0, the processor responds by acknowledging the request by asserting BDIN L and BIAKO L.
AM2 AN2	BIAKI L BIAKO L	Interrupt Acknowledge—In accordance with interrupt protocol, the processor asserts BIAKO L to acknowledge receipt of an interrupt. The bus transmits this to BIAKI L of the

BUS PIN	MNEMONICS	DESCRIPTION
		device electrically closest to the processor. This device accepts the interrupt acknowledge under two conditions: 1) The device requested the bus by asserting BIRQXL, and 2) the device has the highest-priority interrupt request on the bus at that time. If these conditions are not met, the device asserts BIAKO L to the next device on the bus. This process continues in a daisy-chain fashion until the device with the highest-interrupt priority receives the interrupt acknowledge signal.
AP2	BBS7 L	Bank 7 Select—The bus master asserts this signal to reference the I/O page (including that portion of the I/O page reserved for nonexistent memory). The address in BDAL<0:12> L when BBS7 L is asserted is the address within the I/0 page.
AR2 AS2	BDMGI L BDMGO L	Direct Memory Access Grant—The bus arbitrator asserts this signal to grant bus mastership to a requesting device, according to bus mastership protocol. The signal is passed in a daisy-chain from the arbitrator (as BDMGO L) through the bus to BDMGI L of the next priority device (electrically closest device on the bus). This device accepts the grant only if it requested to be bus master (by a BDMR L). If

BUS PIN	MNEMONICS	DESCRIPTION
		not, the device passes the grant (asserts BDMGO L) to the next device on the bus. This process continues until the requesting device acknowledges the grant. **CAUTION** DMA device transfers must not interfere with the memory refresh cycle.
AT2	BINIT L	Initialize—This signal is used for system reset. All devices on the bus are to return to a known, initial state; i.e., registers are reset to zero, and logic is reset to state 0. Exceptions should be completely documented in programing and engineering specifications for the device.
AU2 AV2	BDALO L BDAL1 L	Data/Address lines—These two lines are part of the 16-line data/address bus over which address and data information are communicated. Address information is first placed on the bus by the bus master device. The same device then either receives input data from, or outputs data to the addressed slave device or memory over the same bus lines.
BA2	+5	+5V Power—Normal +5 Vdc system power.
BB2	−12	*−12V Power—−12 Vdc (optional) power for devices requiring this voltage.

* Voltages normally not supplied by DIGITAL.

BUS PIN	MNEMONICS	DESCRIPTION
BC2	GND	Ground—System signal ground and dc return.
BD2	+12	+12V Power— +12V system power.
BE2	BDAL2 L	Data/Address Lines—These 14 lines are part of the 16-line data/address bus previously described.
BF2	BDAL3 L	
BH2	BDAL4 L	
BJ2	BDAL5 L	
BK2	BDAL6 L	
BL2	BDAL7 L	
BM2	BDAL8 L	
BN2	BDAL9 L	
BP2	BDAL10 L	
BR2	BDAL11 L	
BS2	BDAL12 L	
BT2	BDAL13 L	
BU2	BDAL14 L	
BV2	BDAL15 L	

APPENDIX F
INSTRUCTION SET

PDP-11 instructions are represented here in three groups:

Section A: Standard Instructions. Except as noted, these instructions are standard on the F-11-based MICRO/PDP-11 and on the MICRO/J-11.

Section B: Floating-Point Instructions. These are standard on the MICRO/J-11, and optional on the F-11-based MICRO/PDP-11. (Refer to "Data Representations" in Chapter 7.)

Section C: Character and Decimal-String Instructions (Commercial Instruction Set). These are optional on both the F-11-based MICRO/PDP-11 and on the MICRO/J-11. (Refer to "Data Representations" in Chapter 7.)

For additional information on PDP-11 instructions, refer to the *PDP-11 Architecture Handbook*.

SECTION A: STANDARD INSTRUCTIONS

Symbol	Meaning
SO	Single-operand instruction
DO	Double-operand instruction
PC	Program control instruction
MS	Miscellaneous instruction
CC	Condition code
()	Indicates "the contents of"; for example, (R5) means "the contents of R5."
src	Source address (represented in octal instruction format as SS)
dst	Destination address (represented in octal instruction format as DD)

←	Becomes, or moves into; for example, (dst) ← (src) means that the contents of the destination are replaced by the contents of the source.
(SP)+	Popped or removed from the hardware stack
-(SP)	Pushed or added to the hardware stack
AND	Logical AND (both)
OR	Logical inclusive OR (either one or both)
XOR	Logical exclusive OR (either one, but not both)
~	Logical NOT
Reg or R	Register
B	Byte
tmp	Temporary storage location

Note: Condition code bits are considered to be cleared unless they are specifically listed as set.

ADd Carry (Byte) **ADC dst** **$0055DD_8$**

ADCB dst **$1055DD_8$**

Type: SO

Operation: (dst) ← (dst) + C

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if (dst) was 077777_8 and (C) was 1 (ADC); set if (dst) was 177_8 and (C) was 1 (ADCB)
C: set if (dst) was 177777_8 and (C) was 1 (ADC); set if (dst) was 200_8 and (C) was 1 (ADCB)

Description: Adds the contents of the C bit to the destination. This permits the carry from the addition of the low-order words/bytes to be carried into the high-order result, such as in performing double-precision arithmetic.

ADD **ADD src,dst** **$06SSDD_8$**

Type: DO

Operation: (dst) ← (src) + (dst)

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if there is arithmetic overflow as a result of the operation; that is, both operands were of the same sign and the result is of the opposite sign
C: set if there is a carry from the most significant bit of the result

Description: Adds the source operand to the destination operand and stores the result at the destination address. The original contents of the destination are lost. The contents of the source are not affected. Two's complement addition is performed.

Arithmetic SHift **ASH R,src** **$072RSS_8$**

Type: DO

Operation: (R) ← (R) shifted arithmetically NN places to the right or left where NN = (src) <5:0>

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if sign of register changed during shift. Cleared if NN = 0
C: loaded from last bit shifted out of register. Cleared if NN = 0

Description: The contents of the register are shifted right or left the number of times specified by the shift count (the low-order six bits of the source operand). This number ranges from −32 to +31. Negative is a right shift and positive is a left shift.

Arithmetic SHift Combined **ASHC R,src** **$073RSS_8$**

Type: DO

Operation: (R,R OR 1) ← (R,R OR 1)
The double word is shifted NN places to the right or left, where NN = (src)<5:0>.

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if sign bit changes during the shift
C: loaded with the last bit shifted out of the 32-bit operand (high-order bit when left shift; low-order bit when right shift)

Description: The contents of the register R and the register R OR 1 are treated as a single 32-bit operand and are shifted by the number of bits specified by the shift count (the low-order six bits of the source operand). This number ranges from −32 to +31. Negative is a right shift and positive is a left shift. A zero count implies no shift, but condition codes are affected. Condition codes are always set on the 32-bit result.

Notes: (1)The sign bit of the register R is replicated in shifts to the right. The least significant bit is filled with 0 in shifts to the left.

(2) Integer overflow occurs on a left shift if any bit shifted into the sign position differs from the initial sign of the register.

Arithmetic Shift Left (Byte) **ASL dst** **$0063DD_8$**
ASLB dst **$1063DD_8$**

Type: SO

Operation: (dst) ← (dst) shifted one place to the left

Condition Codes:
N: set if high-order bit of the result is set (result < 0)
Z: set if the result = 0
V: loaded with the exclusive OR of the N bit and C bit (as set by the completion of the shift operation)
C: loaded with the high-order bit of the destination

Description: Shifts all bits of the destination left one place. The low-order bit is loaded with a 0. The C bit of the status word is loaded from the high-order bit of the destination. ASL performs a signed multiplication of the destination by 2 with overflow indication.

Arithmetic Shift Right (Byte) **ASR dst** **$0062DD_8$**
ASRB dst **$1062DD_8$**

Type: SO

Operation: (dst) ← (dst) shifted one place to the right

Condition Codes: N: set if the high-order bit of the result is set (result < 0)
Z: set if the result = 0
V: loaded from the exclusive OR of the N bit and C bit (as set by the completion of the shift operation)
C: loaded from low-order bit of the destination

Description: Shifts all bits of the destination right one place. The high-order bit is replicated. The C bit is loaded from the low-order bit of the destination. ASR performs signed division of the destination by 2.

Branch if Carry Clear **BCC address** **103000_8**
+ 8-bit offset

Type: PC

Operation: PC ← PC + (2 X offset) if C = 0

Condition Codes: Unaffected

Description: Tests the state of the C bit and causes a branch if C is clear.

Branch if Carry Set **BCS address** **103400_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if C = 1

Condition Codes: Unaffected

Description: Tests the state of the C bit and causes a branch if C is set. Used to test for a carry in the result of a previous operation.

Branch if EQual (to zero) **BEQ address** **001400_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if Z = 1

Condition Codes: Unaffected

Description: Tests the state of the Z bit and causes a branch if Z is set. As an example, it is used to test equality following a CMP operation, to test that no bits set in the destination were also set in the source following a BIT operation, and, generally, to test that the result of the previous operation was 0.

Branch if Greater than or Equal **BGE address** **002000_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if (N XOR V) = 0

Condition Codes: Unaffected

Description: Causes a branch if N and V are either both clear or both set. BGE is the complementary operation to BLT. Thus, BGE always causes a branch when it follows an operation that caused addition on two positive numbers. BGE also causes a branch on a 0 result.

Branch if Greater Than **BGT address** **003000_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if [Z OR (N XOR V)] = 0

Condition Codes: Unaffected

Description: Causes a branch if Z is clear and N equals V. Thus, BGT never branches after an operation that added two negative numbers, even if overflow occurred. In particular, BGT never causes a branch if it follows a CMP instruction operating on a negative source and a positive destination (even if overflow occurred). Further, BGT always causes a branch when it follows a CMP instruction operating on a positive source and negative destination. BGT does not cause a branch if the result of the previous operation was 0 (without overflow).

Branch if HIgher **BHI address** **101000_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if C = 0 and Z = 0

Condition Codes: Unaffected

Description: Causes a branch if the previous operation caused neither a carry nor a 0 result. This will happen in comparison (CMP) operations as long as the source has a higher unsigned value than the destination.

Branch if Higher than or Same **BHIS address** **103000_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if C = 0

Condition Codes: Unaffected

Description: Tests the state of the C bit and causes a branch if C is cleared.

Bit Clear (Byte) **BIC src,dst** **$04SSDD_8$**

BICB src,dst **$14SSDD_8$**

Type: DO

Operation: (dst) ← ~ (src) AND (dst)

Condition Codes: N: set if high-order bit of result set
Z: set if result = 0
V: cleared
C: unaffected

Description: Clears each bit in the destination that corresponds to a set bit in the source. The original contents of the destination are lost. The contents of the source are unaffected.

Bit Set (Byte) **BIS src,dst** **$05SSDD_8$**

BISB src,dst **$15SSDD_8$**

Type DO

Operation: (dst) ← (src) OR (dst)

Condition Codes: N: set if high-order bit of result set
Z: set if result = 0
V: cleared
C: unaffected

Description: Performs an inclusive OR operation between the source and destination operands and leaves the result at the destination address; i.e., corresponding bits set in the source are set in the destination. The original contents of the destination are lost.

Bit Test (Byte) **BIT src,dst** **$03SSDD_8$**

BITB src,dst **$13SSDD_8$**

Type: DO

Operation: (dst) AND (src)

Condition Codes: N: set if high-order bit of result set
Z: set if result = 0
V: cleared
C: unaffected

Description: Performs a logical AND comparison of the source and destination operands and modifies condition codes accordingly. Neither the source nor destination operands are affected. The BIT instruction may be used to test whether any of the corresponding bits that are set in the destination are clear in the source.

Branch if Less than or Equal to **BLE address** **003400_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if [Z OR (N XOR V)] = 1

Condition Codes: Unaffected

Description: Causes a branch if Z is set or if N does not equal V. Thus, BLE always branches after an operation that added two negative numbers, even if overflow occurred. In particular, BLE always causes a branch if it follows a CMP instruction operating on a negative source and a positive destination (even if overflow occurred). Further, BLE never causes a branch when it follows a CMP instruction operating on a positive source and negative destination. BLE always causes a branch if the result of the previous operation was 0.

Branch if LOwer **BLO address** **103400_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if C = 1

Condition Codes: Unaffected

Description: Tests the state of the C bit and causes a branch if C is set. Used to test for a carry in the result of a previous operation.

Branch if LOwer or Same **BLOS address** **101400_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if (C OR Z) = 1

Condition Codes: Unaffected

Description: Causes a branch if the previous operation caused either a carry or a 0 result. BLOS is the complementary operation to BHI. The branch occurs in comparison operations as long as the source is equal to or has a lower unsigned value than the destination.

Branch if Less Than **BLT address** **002400_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if (N XOR V) = 1

Condition Codes: Unaffected

Description: Causes a branch if the exclusive OR of the N and V bits is 1. Thus, BLT always branches after an operation that added two negative numbers, even if overflow occurred. In particular, BLT always causes a branch if it follows a CMP instruction operating on a negative source and positive destination (even if overflow occurred). Further, BLT never causes a branch when it follows a CMP instruction operating on a positive source and negative destination. BLT does not cause a branch if the result of the previous operation was 0 (without overflow.)

Branch if MInus **BMI address** **100400_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if N = 1

Condition Codes: Unaffected

Description: Tests the state of the N bit and causes a branch if N is set. Used to test the sign (most significant bit) of the result of the previous operation.

Branch if Not Equal **BNE address** **001000_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if Z = 0

Condition Codes: Unaffected

Description: Tests the state of the Z bit and causes a branch if the Z bit is clear. BNE is the complementary operation to BEQ. It is used to test inequality following a CMP, to test that some bits set in the destination were also in the source following a BIT, and, generally, to test that the result of the previous operation was not 0.

Branch if PLus **BPL address** **100000_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if N = 0

Condition Codes: Unaffected

Description: Tests the state of the N bit and causes a branch if N is clear. BPL is the complementary operation of BMI.

BreakPoint Trap **BPT** **000003_8**

Type: PC

Operation: -(SP) ← PS
-(SP) ← PC
PC ← $(14)_8$
PS ← $(16)_8$

Condition Codes: Loaded from trap vector

Description: Performs a trap sequence with a trap vector address of 14_8. Used to call debugging aids. The user is cautioned against employing code 000003_8 in programs run under these debugging aids. No information is transmitted in the low byte.

BRanch (unconditional) **BR address** **000400_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset)

Condition Codes: Unaffected

Description: Provides a way of transferring program control within a range of 128128 to +127 words with a 1-word instruction.

Branch if oVerflow bit Clear **BVC address** **102000_8 +8 bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if V = 0

Condition Codes: Unaffected

Description: Tests the state of the V bit and causes a branch if the V bit is clear. BVC is the complementary operation to BVS.

Branch if oVerflow bit Set **BVS address** **102400_8 + 8-bit offset**

Type: PC

Operation: PC ← PC + (2 X offset) if V = 1

Condition Codes: Unaffected

Description: Tests the state of the V bit (overflow) and causes a branch if the V bit is set. BVS is used to detect arithmetic overflow in the previous operation.

clear selected condition code bits **000240_8 + 4-bit mask**

Type: CC

Operation: PSW <3:0> ← PSW <3:0> AND [~ mask <3:0>]

Description: Clear condition code bits. Selectable combination of these bits may be cleared together. Condition code bits corresponding to bits in the condition code operator (bits 0–3) are modified. Clears bits specified by the mask.

Clear all Condition Code bits **CCC** **000257_8**

Type: CC
Operation: N, Z, V, C ← 0
Description: Clears all condition code bits.

CLear C **CLC** **000241_8**

Type: CC
Operation: C ← 0
Description: Clears condition code C.

CLear N **CLN** **000250_8**

Type: CC
Operation: N ← 0
Description: Clears condition code N.

CLear V **CLV** **000242_8**

Type: CC
Operation: V ← 0
Description: Clears condition code V.

CLear Z **CLZ** **000244_8**

Type: CC
Operation: Z ← 0
Description: Clears condition code Z.

CLeaR (Byte) **CLR dst** **$0050DD_8$**

CLRB dst **$1050DD_8$**

Type: SO

Operation: (dst) ← 0

Condition Codes: N: cleared
Z: set
V: cleared
C: cleared

Description: Contents of specified destination are replaced with 0s.

CoMPare (Byte) **CMP src,dst** **$02SSDD_8$**

CMPB src,dst **$12SSDD_8$**

Type: DO

Operation: (src) – (dst) [in detail (src) + ~(dst) + 1]

Note that unlike subtraction, the order of operation is (src) –(dst), not (dst) – (src).

Condition Codes: N: set if result < 0
Z: set if result = 0
V: set if there is arithmetic overflow; i.e., if the operands were of opposite signs and the sign of the destination is the same as the sign of the result
C: cleared if there is a borrow into the most significant bit of the result
(i.e., if (src) + ~(dst) + 1 was less than 2^{16})

Description: Compares the source and destination operands and sets the condition codes, which may then be used for arithmetic and logical conditional branches. Both operands are unaffected. The only action is to set the condition codes. The compare is customarily followed by a conditional branch instruction.

COMplement (Byte) **COM dst** **$0051DD_8$**

COMB dst **$1051DD_8$**

Type: SO

Operation: (dst) ← ~ (dst)

Condition Codes:
N: set if most significant bit of result = 0
Z: set if result = 0
V: cleared
C: set

Description: Replaces the contents of the destination address by their logical complements (each bit equal to 0 set, and each bit equal to 1 cleared).

Call Supervisor Mode **CSM dst** **$0070DD_8$**

Type: PC

Operation:
If MMR3 <3> = 1 and current mode ≠ Kernel mode
Supervisor SP ← current mode SP
tmp <15:4> ← PSW <15:4>
tmp <3:0> ← 0
PSW <13:12> ← PSW <15:14>
PSW <15:14> ← 01
PSW <4> ← 0
–(SP) ← tmp
–(SP) ← PC
–(SP) ← (dst)
PC ← (10)
else trap to 10 in Kernel mode

Condition Codes: Unaffected

Description: CSM may be executed in user or supervisor mode, but is an illegal instruction in kernel mode. CSM copies the current stack pointer (SP) to the supervisor mode SP, switches to supervisor mode, stacks three words on the supervisor stack, (the PSW with the condition codes cleared, the PC, and the argument word addressed by the operand), and sets the PC to the contents of location 10 (in supervisor space). The called program in supervisor space may return to the

calling program by popping the argument word from the stack and executing RTI. On return, the condition codes are determined by the PSW word on the stack. Hence, the called program in Supervisor space may control the condition code values following return.

Note: Processors without supervisor mode, including the F-11-based MICRO/PDP-11, do not implement the CSM instruction. MICRO/J-11 implements CSM as described.

DECrement (Byte) **DEC dst** **$0053DD_8$**

DECB dst **$1053DD_8$**

Type: SO

Operation: (dst) ← (dst) – 1

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if (dst) was 100000_8 (DEC); set if (dst) was 200_8 (DECB)
C: unaffected

Description: Subtract 1 from the contents of the destination.

DIVide **DIV R,src** **$071RSS_8$**

Type: DO

Operation: (R,R OR 1) ← (R,R OR 1) / (src)

Condition Codes:
N: set if quotient < 0 (unspecified if V = 1)
Z: set if quotient = 0 (unspecified if V = 1)
V: set if (src) = 0 or if quotient cannot be represented as 16-bit 2's complement number
C: set if divide by 0 attempted

Description: The 32-bit 2's complement integer in (R, R OR 1) is divided by the source operand. The quotient is left in R; the remainder is of the same sign as the dividend. R must be even. (R, R OR 1) are unpredictable if V is set and C is clear.

EMulator Trap **EMT** **104000_8**

Type: PC

Operation:
$-(SP) \leftarrow PS$
$-(SP) \leftarrow PC$
$PC \leftarrow (30)_8$
$PS \leftarrow (32)_8$

Condition Codes: Loaded from trap vector

Description: All operation codes from 104000_8 to 104377_8 are EMT instructions and may be used to transmit information to the emulating routine (e.g., function to be performed). The trap vector for EMT is at address 30. The new PC is taken from the word at address 30_8; the new processor status (PS) is taken from the word at address 32_8.

Caution: EMT is used frequently by DIGITAL system software and is therefore not recommended for general use.

HALT **HALT** **000000_8**

Type: MS

Condition Codes: Unaffected

Description: Causes program operation to cease and enters console ODT. (If memory management is present, program execution ceases only if in kernel mode; a trap to location 10_8 occurs if in user mode).

INCrement (Byte) **INC dst** **$0052DD_8$**
INCB dst **$1052DD_8$**

Type: SO

Operation: $(dst) \leftarrow (dst) + 1$

Condition Codes:
N: set if result < 0
Z: set if result $= 0$
V: set if (dst) was 077777_8 (INC);
set if (dst) was 177_8 (INCB)
C: unaffected

Description: Adds 1 to the contents of the destination.

I/O Trap **IOT** **000004_8**

Type: PC

Operation: $-(SP) \leftarrow PS$
$-(SP) \leftarrow PC$
$PC \leftarrow (20)_8$
$PS \leftarrow (22)_8$

Condition Codes: Loaded from trap vector

Description: Performs a trap sequence with a trap vector address of 20_8. Used to call the I/O executive routine IOX in the paper-tape software system and for error reporting in the disk operating system. No information is transmitted in the low byte.

JuMP **JMP dst** **$0001DD_8$**

Type: PC

Operation: $PC \leftarrow (dst)$

Condition Codes: Unaffected

Description: JMP provides more flexible program branching than is provided with the branch instruction. JMP is not limited to $+177_8$ and -200_8 words as are branch instructions. JMP does generate a second word, however, which makes it slower than branch instructions. Control may be transferred to any location in memory (no range limitation) and can be accomplished with the full flexibility of the addressing modes (with the exception of register mode 0). Execution of a jump with mode 0 causes an illegal instruction condition and a trap to location 4_8. (Program control cannot be transferred to a register.) Register-deferred mode is legal and causes program control to be transferred to the address held in the specified register.

Note: Instructions are word data and therefore must be fetched from an even-numbered address. A boundary error trap condition will result when the processor attempts to fetch an instruction from an odd address.

Jump to SubRoutine **JSR R,dst** **$004RDD_8$**

Type: PC

Operation: (tmp) ← (dst)(tmp is an internal processor register)
-(SP) ← (R) (push reg contents onto processor stack)
(R) ← PC (PC holds location following JSR; this address now put in reg)
PC ← (tmp) (PC now points to subroutine address)

Condition Codes: Unaffected

Description: In execution of the JSR, the old contents of the specified register (the linkage pointer) are automatically pushed onto the processor stack and new linkage information placed in the register. Thus, subroutines nested within subroutines to any depth may all be called with the same linkage register. There is no need either to plan the maximum depth at which any particular subroutine will be called or to include instructions in each routine to save and restore the linkage pointer. Further, since all linkages are saved in a reentrant manner on the processor stack, execution of a subroutine may be interrupted, and the same subroutine reentered and executed by an interrupt service routine. Execution of the initial subroutine can then be resumed when other requests are satisfied. This process (called nesting) can proceed to any level.

JSR PC,dst is a special case of the PDP-11 subroutine call suitable for subroutine calls that transmit parameters through the general registers. JSR with the PC as the linkage register saves the use of an extra register.

A JSR with mode 0 will result in an illegal instruction and a trap through the trap vector address 4.

Note: If the register specified in the first operand (src) is autoincremented or autodecremented in the second operand (dst) evaluation, the modified register content is pushed on SP. For example, JSR R5, @ (R5) + will cause the modified value of R5 to be pushed on SP.

MARK **MARK number** **$0064NN_8$**

Type: PC

Operation: SP ← PC + 2 X NN
PC ← R5
R5 ← (SP)+
NN = number of parameters

Condition Codes: Unaffected

Description: Used as part of the standard PDP-11 subroutine return convention. MARK facilitates the stack clean-up procedures involved in subroutine exit.

Move From Previous Data space **MFPD src** **$0065SS_8$**

Move From Previous Instruction space **MFPI src** **$1065SS_8$**

Type: MS

Operation: (tmp) ← (src)
–(SP) ← (tmp)

Condition Codes: N: set if the source < 0
Z: set if the source = 0
V: cleared
C: unaffected

Description: Pushes a word onto the current R6 stack from an address in the previous space determined by PS <13:12>. The source address is computed using the current registers and memory map. When MFPI is executed and both previous mode and current mode are User, the instruction functions as though it were MFPD.

Note: Processors without data space, including the F-11-based MICRO/PDP-11, execute MFPD the same as MFPI. MICRO/J-11 implements MFPD as described.

Move From Processor Status **MFPS dst** **$1067DD_8$**

Type: MS

Operation: (dst) ← PS<7:0>
dst lower 8 bits

Condition Codes: N: set if PS bit 7 = 1
Z: set if PS <0:7> = 0
V: cleared
C: unaffected

Description: The 8-bit contents of the PS are moved to the effective destination. If destination mode is 0, PS bit 7 is sign-extended through the upper byte of the register. The destination operand is treated as a byte address.

Move From Processor Type **MFPT** **000007_8**

Type: MS

Operation: R0<7:0> ← processor model code
R0<15:8> ← processor subcode

Condition Codes: Unaffected

Description: A unique number assigned to each PDP-11 processor model is loaded into general register R0. No source operands are used. MFPT returns in the low byte of RO the processor model code. The high byte of RO is loaded with a processor-specific subcode. The previous contents of RO are lost.

Note: On processors where this instruction is not implemented, a reserved instruction trap through vector 10_8 is taken. The F-11-based MICRO/PDP-11 and MICRO/J-11 both implement MFPT.

MOVe (Byte) **MOV src,dst** **$01SSDD_8$**

MOVB src,dst **$11SSDD_8$**

Type: DO

Operation: (dst) ← (src)

Condition Codes:
N: set if (src) < 0
Z: set if (src) = 0
V: cleared
C: unaffected

Description: Moves the source operand to the destination location. The previous contents of the destination are lost. The source operand is not affected.

Byte: Same as MOV. The MOVB to a register (mode 0), which is unique among byte instructions, extends the most significant bit of the low-order byte (sign extension) into the high byte of the selected register. Otherwise, MOVB operates on bytes exactly as MOV operates on words.

Move To Previous Data space **MTPD src** **$1066SS_8$**

Move To Previous Instruction space **MTPI src** **$0066SS_8$**

Type: MS

Operation: (tmp) ← (SP)+
(dst) ← (tmp)

Condition Codes:
N: set if the source < 0
Z: set if the source = 0
V: cleared
C: unaffected

Description: This instruction pops a word off the current stack determined by PS <15:14> and stores that word into an address in the previous space determined by PS <13:12>. The destination address is computed using the current registers and memory map.

Note: Processors without data space, including the F-11-based MICRO/PDP-11, execute MTPD the same as MTPI. MICRO/J-11 implements MTPD as described.

Move byte To Processor Status word **MTPS src** **$1064SS_8$**

Type: MS

Operation: PS ← (src)

Condition Codes: Set according to effective src operand bits 0–3

Description: The eight bits of the effective operand replace the current contents of PS <7:0> if in kernel mode. Only PS bits 0 through 3 are affected if in user mode. The source operand address is treated as a byte address. Note that PS bit 4 (T bit) cannot be set with this instruction in either kernel or user mode. The src operand remains unchanged.

MULtiply **MUL R,src** **$070RSS_8$**

Type: DO

Operation: (R, R OR 1) ← (R) X (src)

Condition Codes:
N: set if product < 0
Z: set if product = 0
V: cleared
C: set if the result is less than -2^{15} or greater than or equal to 2^{15-1}

Description: The contents of the destination register and source taken as 2's complement integers are multiplied and stored in the destination register and the succeeding register if R is even. If R is odd, only the low-order product is stored. [Note that the actual destination is (R, R OR 1) which reduces to just R when R is odd.]

NEGate (Byte) **NEG dst** **$0054DD_8$**

NEGB dst **$1054DD_8$**

Type: SO

Operation: (dst) ← −(dst)

Condition Codes: N: set if result < 0
Z: set if result = 0
V: set if result = 100000_8 (NEG);
set if result = 200_8 (NEGB)
C: cleared if result = 0; set otherwise

Description: Replaces the contents of the destination address by its 2's complement. Note that 100000_8 (NEG) or 200_8 (NEGB) is replaced by itself.

No OPeration **NOP** **000240_8**

NOP **000260_8**

Type: CC

Operation: None

Condition Codes: Unaffected

Description: No operation is performed.

RESET **RESET** **000005_8**

Type: MS

Condition Codes: Unaffected

Description: Causes bus signal BINIT L to be asserted for 10 mLs and then unasserted for 90 mLs. Used to initialize I/O devices attached to the bus. In addition, memory management status registers SR0 and SR3 are cleared.

ROtate Left (Byte) **ROL dst** **$0061DD_8$**

ROLB dst **$1061DD_8$**

Type: SO

Operation: (dst) ← (dst)
rotate left one place

Condition Codes: N: set if the high-order bit of the result word is set (result < 0)
Z: set if all bits of the result word = 0
V: loaded with the N XOR C (as set by the completion of the rotate operation)
C: set if the high-order bit of the destination was set prior to instruction execution

Description: Rotates all bits of the destination left one place. The high-order bit is loaded into the C bit of the status word and the previous contents of the C bit are loaded into the low-order bit of the destination.

ROtate Right **ROR dst** **$0060DD_8$**

RORB dst **$1060DD_8$**

Type: SO

Operation: (dst) ← (dst)
rotate right one place

Condition Codes: N: set if high-order bit of the result is set (result < 0)
Z: set if all bits of result are 0
V: loaded with the N XOR C (as set by the completion of the rotate operation)
C: set if the low-order bit of the destination was set prior to instruction execution

Description: Rotates all bits of the destination right one place. The low-order bit is loaded into the C bit and the previous contents of the C bit are loaded into the high-order bit of the destination.

ReTurn from Interrupt **RTI** **000002_8**

Type: MS

Operation: PC ← (SP) +
PS ← (SP) +

Condition Codes: Loaded from current R6 stack

Description: Used to exit from an interrupt or trap service routine. The PC and PS are restored (popped) from the processor stack. If the RTI sets the T bit in the PS, a trace trap will occur prior to executing the next instruction. When executed in user mode, the current and previous mode bits in the restored PS can only be User. RTI cannot clear PS <11> if it was already set. When executed in user mode, PS <7:5> are unaffected.

ReTurn from Subroutine **RTS R** **$00020R_8$**

Type: PC

Operation: PC ← (reg)
(reg) ← SP+

Condition Codes: Unaffected

Description: Loads the contents of the register into the PC and pops the top element of the processor stack into the specified register. Return from a nonreentrant subroutine is typically made through the same register that was used in its call. Thus, a subroutine called with a *JSR PC,dst* exits with an *RTS PC*, and a subroutine called with a *JSR R5,dst* may pick up parameters with addressing modes (R5)+, X(R5), or @X(R5) and finally exit with an *RTS R5.*

ReTurn from Trap **RTT** **000006_8**

Type: MS

Operation: PC ← (SP)+
PS ← (SP)+

Condition Codes: Loaded from current R6 stack

Description: Used to exit from a trace trap (T bit) service routine and executes in the same way as the RTI instruction does, with one exception. If the RTT sets the T bit in the PS, the next instruction will be executed and then the trace trap will be processed. However, if an RTI sets the T bit in the PS, a trace trap will occur before the next instruction is executed.

SuBtract Carry (Byte) **SBC dst** **$0056DD_8$**

SBCB dst **$1056DD_8$**

Type: SO

Operation: (dst) ← (dst) - C

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if (dst) was 100000_8 and C was 1 (SBC); set if (dst) was 200_8 and C was 1 (SBCB)
C: cleared if (dst) was 0 and C was 1

Description: Subtract the contents of the C bit from the destination. This permits the carry from the subtraction of the low-order words/bytes to be subtracted from the high-order part of the result in order to perform double-precision subtraction.

set selected condition codes **000260 + 4-bit mask**

Type: CC

Operation: PSW <3:0> ← PSW <3:0> OR mask <3:0>

Description: Sets condition code bits. Selectable combinations of these bits may be set together. Condition codes corresponding to bits in the condition code operator (bits 0–3) are modified; sets the bits specified by the mask.

Set all Condition Code bits **SCC** **000277_8**

Type: CC

Operation: N, Z, V, C ← 1

Description: Sets all condition code bits.

SEC **SEC** **000261_8**

Type: CC

Operation: C ← 1

Description: Sets condition code C.

SEN **SEN** **000270_8**

Type: CC

Operation: N ← 1

Description: Sets condition code N.

SEV **SEV** **000262_8**

Type: CC

Operation: V ← 1

Description: Sets condition code V.

SEZ **SEZ** **000264_8**

Type: CC

Operation: Z ← 1

Description: Sets condition code Z.

Set Priority Level **SPL N** **$00023N_8$**

Type: PC

Operation: PS <7:5> ← N (priority)

Condition Codes: Unaffected

Description: The least significant bits of the instruction are loaded into the PSW bits 7–5, causing a changed priority. The old priority is lost.

Note: Not implemented in F-11-based MICRO/PDP-11. MICRO/J-11 implements SPL as described.

Subtract One and Branch if not equal to 0 **SOB R,address** **$077R00_8$ + 6-bit offset**

Type: PC

Operation: (R) ← (R) – 1; if this result does not = 0 then PC ← PC – (2 X offset)

Condition Codes: Unaffected

Description: The register is decremented. If it is not equal to 0, twice the offset is subtracted from the PC (now pointing to the following word). The offset is interpreted as a 6-bit positive number. This instruction provides a fast, efficient method of loop control. Note that the SOB instruction cannot be used to transfer control in the forward direction.

SUBtract **SUB src,dst** **$16SSDD_8$**

Type: DO

Operation: (dst) ← (dst) – (src)
[in detail (dst) ← (dst)+~(src) + 1]

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if there is arithmetic overflow as a result of the operation, i.e., if the operands were of opposite signs and the sign of the source is the same as the sign of the result
C: cleared if there is a borrow into the most significant bit of the result (i.e., if (dst)+~(src) + 1 was less than 2^{16})

Description: Subtracts the source operand from the destination operand and leaves the result at the destination address. The original contents of the destination are lost. The contents of the source are not affected. For double-precision arithmetic, the C bit indicates a borrow when set.

SWAp Byte **SWAB dst** **$0003DD_8$**

Type: SO

Operation: tmp ← (dst) <7:0>
(dst) <7:0> ← (dst) <15:8>
(dst) <15:8> ← tmp

Condition Codes:
N: set if high-order bit of low-order byte (bit 7) of result is set
Z: set if low-order byte of result = 0
V: cleared
C: cleared

Description: Exchanges the high-order byte and low-order byte of the destination, which must be a word address.

Sign eXTend **SXT dst** **$0067DD_8$**

Type: SO

Operation: (dst) ← 0 if N bit is clear
(dst) ← −1 if N bit is set

Condition Codes: N: unaffected
Z: set if N bit clear
V: cleared
C: unaffected

Description: If the condition code bit N is set, a −1 is placed in the destination operand; if N bit is clear, a 0 is placed in the destination operand. This instruction is particularly useful in multiple-precision arithmetic because it permits the sign to be extended through multiple words.

TRAP **TRAP** **104400_8–104777_8**

Type: PC

Operation: −(SP) ← PS
−(SP) ← PC
PC ← (34)
PS ← (36)

Condition Codes: Loaded from trap vector

Description: Operation codes from 104400_8 to 104777_8 are TRAP instructions. TRAPs and EMTs are identical in operation, except that the trap vector for TRAP is at address 34_8.

Note: Since DIGITAL software makes frequent use of EMT, the TRAP instruction is recommended for general use.

TeST (Byte) **TST dst** **$0057DD_8$**

TSTB dst **$1057DD_8$**

Type: SO

Operation: (dst) ← (dst)

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: cleared
C: cleared

Description: Sets the condition codes N and Z according to the contents of the destination address.

TeST destination and SET bit **TSTSET dst** **$0072DD_8$**

Type: SO

Operation:
(RO) ← (dst)
(dst) ← [(RO) OR 1]

Condition Codes:
N: set if RO < 0
Z: set if RO = 0
V: cleared
C: set if low-order bit is 1, cleared otherwise

Description: Reads/locks destination word and stores it in RO. Writes/unlocks [(RO) OR 1] into destination. If mode is 0, traps to 10_8.

Note: Not implemented on F-11-based MICRO/PDP-11. MICRO/J-11 implements TSTSET as described.

WAIT **WAIT** **000001_8**

Type: MS

Condition Codes: Unaffected

Description: Provides a way for the processor to relinquish use of the bus while it waits for an external interrupt. Having been given a WAIT command, the processor will not compete for the bus by fetching instructions or operands from memory. This permits higher transfer

rates between device and memory since no processor-induced latencies will be encountered by bus requests from the device. In WAIT, as in all instructions, the PC points to the next instruction following the WAIT operation. Thus, when an interrupt causes the PC and PS to be pushed onto the stack, the address of the next instruction following the WAIT is saved. The exit from the interrupt routine (i.e., execution of an RTI instruction) will cause resumption of the interrupted process at the instruction following the WAIT.

read/lock destination **WRTLCK dst** **$0073DD_8$**

WRiTe/unLoCK R0

Type: SO

Operation: (dst) ← (RO)

Condition Codes: N: set if R0 < 0
Z: set if R0 = 0
V: clear
C: unaffected

Description: Writes contents of R0 into destination using bus lock. If mode is 0, traps to 10_8.

Note: Not implemented on F-11-based MICRO/PDP-11. MICRO/J-11 implements WRTLCK as described.

eXclusive OR **XOR R,dst** **$074RDD_8$**

Type: DO

Operation: (dst) ← (R) XOR (dst)

Condition Codes: N: set if the result < 0
Z: set if result = 0
V: cleared
C: unaffected

Description: The exclusive OR of the register and destination operand is stored in the destination address. The contents of the register are unaffected.

SECTION B: FLOATING-POINT INSTRUCTIONS

FLOATING-POINT INSTRUCTIONS

Each instruction that manipulates a floating-point number can operate on either single-precision or double-precision numbers, depending on the state of FD mode bit. Similarly, there is a mode bit FL that determines whether 32-bit integers or 16-bit integers are used in conversion between integer and floating-point representation. In our notation, FSRC and FDST use floating-point addressing modes; SRC and DST use CPU addressing modes. Figure F-1 illustrates single-floating point and double-floating-point operand addressing.

In the descriptions of the floating-point instructions, all implementations operate identically, except where explicitly stated otherwise. Table F-1 describes the floating-point conventions used in the PDP-11 instruction set.

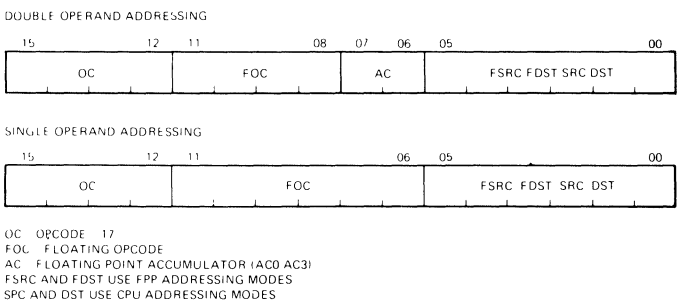


Figure F-1 Single-Operand and Double-Operand Addressing

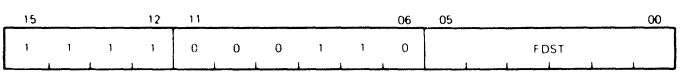
Table F-1 Floating-Point Conventions

Symbolic	Description
OC	Opcode = 17
FOC	Floating Opcode
AC	Contents of accumulator, as specified by AC field of instruction
fsrc	Address of floating-point source operand.
fdst	Address of floating-point destination operand

Symbolic	Description
f	Fraction
XL	Largest fraction that can be represented: 1 – 2**(– 24), FD = 0, single-precision 1 – 2**(– 56), FD = 1; double-precision
XLL	Smallest number that is not identically zero = 2**(– 128)
XUL	Largest number that can be represented = 2**(127)*XL
JL	Largest integer that can be represented: 2**(15) – 1 if FL = 0, 2**(31) – 1 if FL = 1
ABS[(x)]	Absolute value of contents of memory location X
EXP[(x)]	Biased exponent of contents of memory location X
<	Less than
≤	Less than or equal to
>	Greater than
≥	Greater than or equal to
≠	Not equal to
LSB	Least significant bit

Floating-Point Instructions

ABSF
ABSD
Take Absolute Value 1706 FDST



Format: ABSF FDST

Operation: If (fdst) < 0, (fdst) ◂— – (fdst).
If EXP[(fdst)] = 0, (fdst) ◂— exact 0.
For all other cases, (fdst) ◂— (fdst).

Condition Codes: FC ◂— 0
FV ◂— 0
FZ ◂— 1 if (fdst) = 0, else FZ ◂— 0
FN ◂— 0

Description: Set the contents of fdst to its absolute value.

Interrupts: If FIUV is enabled, trap on " – 0" occurs after execution.

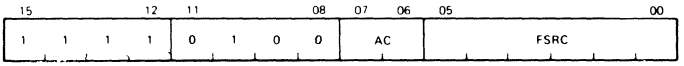
Overflow and underflow cannot occur.

Accuracy: These instructions are exact.

Special Comment: If a " – 0" is present in memory and the FIUV bit is enabled, then an exact zero is stored in memory. The condition codes reflect an exact zero (FZ ◂— 1).

ADDF
ADDD

Add Floating/Double 172(AC)FSRC



Format: ADDF FSRC,AC

Operation: Let SUM = AC + (fsrc). If underflow occurs and FIU is not enabled, AC ◂— exact 0.

If overflow occurs and FIV is not enabled, AC ◂— exact 0.

For all other cases, AC ◂— SUM.

Condition Codes: FC ◂— 0
FV ◂— 1 if overflow occurs, else FV ◂— 0
FZ ◂— 1 if AC = 0, else FZ ◂— 0
FN ◂— 1 if AC < 0, else FN ◂— 0

Description: Add the contents of fsrc to the contents of AC. The addition is carried out in single-precision or double-precision and is rounded or chopped according to the values of the FD and FT bits in the FPS register. The result is stored in AC except for:

- Overflow with interrupt disabled
- Underflow with interrupt disabled.

For these exceptional cases, an exact zero is stored in AC.

Interrupts:

If FIUV is enabled, trap on " −0" in fsrc occurs before execution.

If overflow or underflow occurs and if the corresponding interrupt is enabled, the trap occurs with the faulty result in AC. The fractional parts are correctly stored. The exponent part is too small by 400_8 for overflow. It is too large by 400_8 for underflow, except for the special case of 0, which is correct.

Accuracy:

Errors due to overflow and underflow are described above. If neither occurs, then for oppositely signed operands with an exponent difference of 0 or 1, the answer returned is exact if a loss of significance of one or more bits can occur. Note that these are the only cases for which loss of significance of more than one bit can occur. For all other cases the result is inexact with error bounds of:

- One LSB in truncated mode with either single-precision or double-precision.
- From ½ LSB to ¾ LSB in rounding mode, depending on floating-point option. See Appendix B—PDP-11 Family Differences—for details.

Special Comment:

The undefined variable " −0" can occur only in conjunction with overflow or underflow. It will be stored in AC, only if the corresporflow, except for the special case of 0, which is correct.

Accuracy:

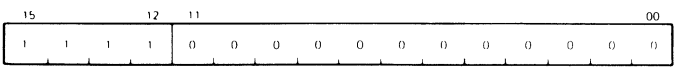
Errors due to overflow and underflow are described above. If neither occurs, then for oppositely signed operands with an exponent difference of 0 or 1, the answer returned is exact if a

loss of significance of one or more bits can occur. Note that these are the only cases for which loss of significance of more than one bit can occur. For all other cases the result is inexact with error bounds of:

- One LSB in truncated mode with either single-precision or double-precision.
- From ½ LSB to ¾ LSB in rounding mode depending on floating-point option. See Appendix B—PDP-11 Family Differences—for details.

Special Comment: The undefined variable " – 0" can occur only in conjunction with overflow or underflow. It will be stored in AC, only if the corresponding interrupt is enabled.

CFCC
Copy Floating Condition Codes 170000

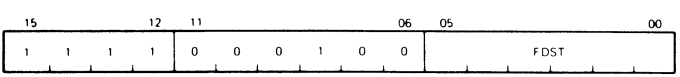


Format: CFCC

Operation:
C ← FC
V ← FV
Z ← FZ
N ← FN

Description: Copy the floating-point condition codes into the CPU's condition codes.

CLRF
CLRD
Clear Floating/Double 1704 FDST



Format: CLRF FDST

Operation: (fdst) ◂— exact 0

Condition Codes: FC ◂— 0
FV ◂— 0
FZ ◂— 1
FN ◂— 0

Description: Set (fdst) to 0. Set FZ condition code, clear other condition code bits.

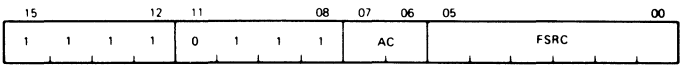
Interrupts: No interrupts will occur.

Overflow and underflow cannot occur.

Accuracy: The instructions are exact.

CMPF
CMPD

Compare Floating/Double 173(AC + 4)FSRC



Format: CMPF FSRC,AC

Operation: (fsrc) – AC

Condition Codes: FC ◂— 0
FV ◂— 0
FZ ◂— 1 if (fsrc) = 0, else FZ ◂— 0
FN ◂— 1 if (fsrc) < 0, else FN ◂— 0

Description: Compare the contents of (fsrc) with the accumulator. Set the appropriate floating-point condition codes. The accumulator and (fsrc) are left unchanged
except as noted below.

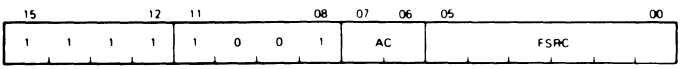
Interrupts: If FIUV is enabled, trap on " – 0" in (fsrc) occurs before execution.

Accuracy: These instructions are exact.

Special Comment: An operand which has a biased exponent of 0 is treated as if it were an exact zero. In this case, where both operands are zero, the FPP will store an exact zero in AC.

DIVF
DIVD

Divide Floating/Double 174(AC + 4)FSRC



Format: DIVF FSRC,AC

Operation: If EXP[(fsrc)] = 0, AC ◂— AC and the instruction is aborted.

If EXP [AC] = 0, AC ◂— exact 0.

For all other cases, let QUOT = AC/(fsrc).

If underflow occurs and FIU is not enabled, AC ◂— exact 0.

If overflow occurs and FIV is not enabled, AC ◂— exact 0.

For all other cases, AC ◂— QUOT.

Condition Codes: FC ◂— 0
FV ◂— 1 if overflow occurs, else FV ◂— 0
FZ ◂— 1 if AC = 0, else FZ ◂— 0
FN ◂— 1 if AC < 0, else FN ◂— 0

Description: If either operand has a biased exponent of zero, it is treated as an exact zero. For fsrc this would imply division by zero; in this case the instruction is aborted, the FEC register is set to four and an interrupt occurs. Otherwise the quotient is developed to single or
double precision with two guard bits for correct rounding. The quotient is rounded and chopped according to the values of the FD and FT bits in the FPS register. The result is stored in the AC except for:

- Overflow with interrupt disabled
- Underflow with interrupt disabled

For these exceptional cases, an exact zero is stored in AC.

Interrupts: If FIUV is enabled, trap on " −0" in (fsrc) occurs before execution.

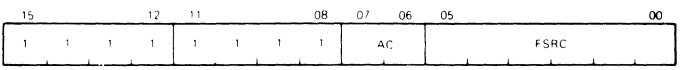
If (fsrc) = 0, interrupt traps on attempt to divide by 0.

If overflow or underflow occurs and if the corresponding interrupt is enabled, the trap occurs with the faulty result in AC. The fractional parts are correctly stored. The exponent part is too small by 400_8 for overflow. It is too large by 400_8 for underflow, except for the special case of 0, which is correct.

Accuracy: Errors due to overflow and underflow are described above. If none of these occur, the error in the quotient will be bounded by one LSB in chopping mode and by ½ LSB in rounding mode.

Special Comment: The undefined variable " −0" can occur only in conjunction with overflow and underflow. It will be stored in AC, only if the corresponding interrupt is enabled.

LDCDF
LDCFD
Load and Convert from Double to Floating and from Floating to Double 177(AC + 4)FSRC



Format: LDCDF FSRC,AC

Operation: If EXP[(fsrc)] = 0, AC ◂— exact 0.

If FD = 1, FT = 0, FIV = 0 and rounding causes overflow, AC ◂— exact 0.

In all other cases, AC ◂— Cxy[(fsrc)], where Cxy specifies conversion from floating mode x to floating mode y.

x = D, y = F if FD = 0 (single) LDCDF
x = F, y = D if FD = 1 (double) LDCFD

Condition Codes: FC ← 0
FV ← 1 if conversion produces overflow, else FV ← 0
FZ ← 1 if AC = 0, else FZ ← 0
FN ← 1 if AC < 0, else FN ← 0

Description: If the current mode is floating mode (FD = 0), the source is assumed to be a double-precision number and is converted to single precision. If the floating chop bit (FT) is set, the number is chopped, otherwise the number is rounded.

If the current mode is double mode (FD = 1), the source is assumed to be a single-precision number and is loaded left-justified into AC. The lower half of AC is cleared.

Interrupts: If FIUV is enabled, the trap on " – 0" occurs before execution. However, the condition codes will reflect a fetch of " – 0" regardless of the FIUV bit.

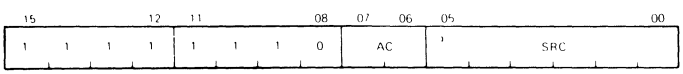
Overflow cannot occur for LDCFD.

A trap occurs if FIV is enabled, and if rounding with LDCDF causes overflow. AC ← overflowed result. This result must be + 0 or " – 0."

Underflow cannot occur.

Accuracy: LDCFD is an exact instruction. Except for overflow, described above, LDCDF incurs an error bounded by one LSB in chopping mode and by LSB in rounding mode.

LDCIF LDCLF
LDCID LDCLD
Load and Convert Integer or Long Integer to Floating or Double-Precision 177(AC)SRC



Format: LDCIF SRC,AC

Operation: AC ← Cjx[(src)], where Cjx specifies conversion from integer mode j to floating mode y.

j = I if FL = 0, j = L if FL = 1
x = F if FD = 0, x = D if FD = 1

Condition Codes:
FC ← 0
FV ← 0
FZ ← 1 if AC = 0, else FZ ← 0
FN ← 1 if AC < 0, else FN ← 0

Description: Conversion is performed on the contents of SRC from a 2's complement integer with precision j to a floating-point number of precision x. Note that j and x are determined by the state of the mode bits FL and FD.

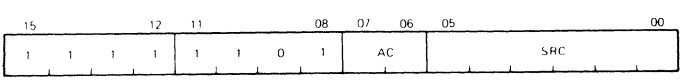
If a 32-bit integer is specified (L mode) and SRC has an addressing mode of 0 or immediate addressing mode is specified, the 16 bits of the source register are left-justified, and the remaining 16 bits are loaded with 0s before conversion.

In the case of LDCLF, the fractional part of the floating-point representation is chopped or rounded to 24 bits according to the state of FT (1 = chop, 0 = round).

Interrupts: None; (SRC) is not floating-point, so trap on "−0" cannot occur.

Accuracy: LDCIF, LDCID, and LDCLD are exact instructions. The error incurred by LDCLF is bounded by one LSB in chopping mode and by ½ LSB in rounding mode.

LDEXP
Load Exponent 176(AC+4)SRC



Format: LDEXP SRC,AR

Operation: If $-200_8 < (src) < 200_8$, EXP[AC] ◂— SRC + 200_8 and the rest of AC is unchanged.

If (src) > 177_8 and FIV is enabled, EXP[AC] ◂— [(src) + 200_8]<7:0> on the FP11-A, -F and KEF11-AA. See Appendix B—PDP-11 Family Differences—for the FP11-C.

If (src) > 177_8 and FIV is disabled, AC ◂— exact 0.

If (src) < -177_8 and FIU is enabled, EXP[AC] ◂— [(src) + 200_8] <7:0> on the FP11-A, -F and KEF11-AA. See Appendix B—PDP-11 Family Differences—for the FP11-C.

If (src) < -177_8 and FIU is disabled,AC ◂— exact 0.

Condition Codes: FC ◂— 0
FV ◂— 1 if (SRC) > 177_8, else FV ◂— 0
FZ ◂— 1 if (AC) = 0, else FZ ◂— 0
FN ◂— 1 if (AC) < 0, else FN ◂— 0

Description: Change AC so that its unbiased exponent equals (src). That is, convert (src) from 2's complement to excess 200_8 notation and insert it in the EXP field of AC. This is a meaningful operation only if ABS[(src)] ≤ 177_8.

If (src) > 177_8, the result is treated as overflow. If (src) < -177_8, the result is treated as underflow. See Appendix B—PDP-11 Family Differences—for treatment of abnormal conditions by the FP-11C and FP-11B.

Interrupts: No trap on "−0" in AC occurs, even if FIUV is enabled.

If (src) > 177_8 and FIV is enabled, trap on overflow will occur.

If (src) < -177_8 and FIU is enabled, trap on underflow will occur.

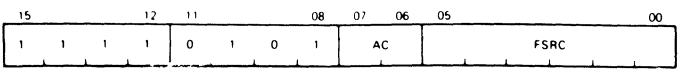
Accuracy: Errors due to overflow and underflow are described above. If EXP[AC] = 0 and (src) ≠ −200, AC changes from a floating-point number treated as zero by all floating arithmetic operations

to a nonzero number. This is because the insertion of the "hidden" bit in the microcode implementation of arithmetic instructions is triggered by a nonvanishing value of EXP.

For all other cases, LDEXP implements exactly the transformation of a floating-point number (2**K) * f into (2**(src)) * f where ½ ≤ ABS(f) < 1.

LDF
LDD

Load Floating/Double 172(AC + 4)FSRC



Format: LDF FSRC,AC

Operation: AC ← (fsrc)

Condition Codes: FC ← 0
FV ← 0
FZ ← 1 if AC = 0, else FZ ← 0
FN ← 1 if AC < 0, else FN ← 0

Description: Load single-precision or double-precision number into AC.

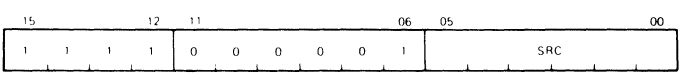
Interrupts: If FIUV is enabled, trap on " – 0" occurs before AC is loaded. However, the condition codes will reflect a fetch " – 0" regardless of the FIUV bit.

Overflow and underflow cannot occur.

Accuracy: These instructions are exact.

Special Comment: These instructions permit use of " – 0" in a subsequent floating-point instruction if FIUV is not enabled and (fsrc) = – 0.

LDFPS
Load FPP Program Status 1701 SRC



Format: LDFPS SRC

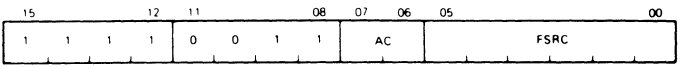
Operation: FPS ◂— (src)

Description: Load FPP status register from (src).

Special Comment: Bits 13, 12, and 4 should not be used for the user's own purposes, since these bits are not recoverable by the STFPS instruction. Bit 4 may be set in Kernel mode if the FPP implements maintenance mode.

MODF
MODD

Multiply and Separate Integer and Fraction Floating/Double 171(AC + 4)FSRC



Format: MODF FSRC,AC

Description and Operation: This instruction generates the product of its two floating point operands, separates the product into integer and fractional parts, and then stores one or both parts as floating-point numbers.

Let PROD = AC * (fsrc) so that in
Floating-point: ABS [PROD] = (2**K) * f
where
½ ≤ f < 1 and
EXP[PROD] = (200 + K) octal
Fixed point binary: PROD = N + g with
N = INT[PROD] = the integer part of PROD and
g = PROD − INT[PROD] = the fractional part of PROD with 0 ≤ g < 1

Both N and g have the same sign as PROD. They are returned as follows:

If AC is an even-numbered accumulator (0 or 2), N is stored in AC + 1 (1 or 3), and g is stored in AC.

If AC is an odd-numbered accumulator, N is not stored, and g is stored in AC.

The two statements above can be combined as follows:

N is returned to ACv1 and g is returned to AC, where v means OR.

Five special cases occur, as indicated in the following formal description with L = 24 for floating mode and L = 56 for double mode.

1. If PROD overflows and FIV is enabled, ACv1 ◂— N, chopped to L bits, AC ◂— exact 0.

Note that EXP[N] is too small by 400_8 and that "−0" can get stored in ACv1.

If FIV is not enabled, ACv1 ◂— exact 0, AC ◂— exact 0, and "−0" will never be stored.

2. If 2**L ≤ ABS[PROD] and no overflow, ACv1 ◂— N, chopped to L bits, AC ◂— exact 0.

The sign and EXP of N are correct, but low-order bit information is lost.

3. If 1 ≤ ABS[PROD] < 2**L, ACv1 ◂— N, AC ◂— g

The integer part N is exact. The fractional part g is normalized, and chopped or rounded in accordance with FT. Rounding may cause a return of ± unity for the fractional part. For L = 24, the error in g is bounded by one LSB in chopping mode and by ½ LSB in rounding mode. For L = 56, the error in g increases from the above limits as ABS[N] increases above 2**L because only 59 bits (64 bits for KEF11-AA) of PROD are generated.

If 2**p ≤ ABS[N] < 2**(p**1), with p > 2 (7 for KEF11-AA) the low-order p − 2 (p − 7 for KEF11-AA) bits of g may be in error.

4. If ABS[PROD] < 1 and no underflow, ACv1 ◂— exact 0 and AC ◂— g.

There is no error in the integer part. The error in the fractional part is bounded by one LSB in chopping mode and ½ LSB in round-

ing mode. Rounding may cause a return of ± unity for the fractional part.

5. If PROD underflows and FIU is enabled, ACv1 ◂— exact 0 and AC ◂— g.

Errors are as in case 4, except that EXP[AC] will be too large by 400_8 (if EXP = 0, it is correct). Interrupt will occur, and "−0" can be stored in AC.

If FIU is not enabled, ACv1 ◂— exact 0 and AC ◂— exact 0.

For this case the error in the fractional part is less than 2**(−128).

Condition Codes:

FC ◂— 0
FV ◂— 1 if PROD overflows, else FV ◂— 0
FZ ◂— If AC = 0, else FZ ◂— 0
FN ◂— 1 if AC < 0, else FN ◂— 0

Interrupts:

If FIUV is enabled, trap on "−0" in FSRC occurs before execution.

Overflow and underflow are discussed above.

Accuracy:

Discussed above.

Applications:

1. Binary-to-decimal conversion of a proper fraction. The following algorithm, using MOD, will generate decimal digits D(1), D(2)... from left to right.

```
Initialize:          I ◂— 0;
X ◂— number to
be converted;
ABS[X] < 1;
While X ≠0 do
Begin PROD ◂— X * 10;
I ◂— I + 1;
D (I) ◂— INT(PROD);
X ◂— PROD − INT(PROD);
End;
```

This algorithm is exact. It is case 3 in the description because the number of nonvanishing bits in the fractional part of PROD never exceeds L, and hence, neither chopping nor rounding can introduce error.

2. To reduce the argument of a trigonometric function.

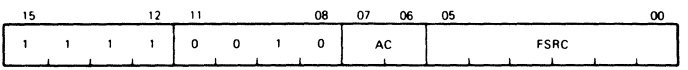
ARG * 2/PI = N + g. The low two bits of N identify the quadrant, and g is the argument reduced to the first quadrant. The accuracy of N + g is limited to L bits because of the factor 2/PI. The accuracy of the reduced argument thus depends on the size of N.

3. To evaluate the exponential function e**x, obtain x * (log e base 2) = N + g, then e**x =
(2**N) * (e**(g*ln 2)).

The reduced argument is g * ln2 < 1 and the factor 2**N is an exact power of two, which may be scaled in at the end via STEXP, ADD N to EXP and LDEXP. The accuracy of N + g is limited to L bits because of the factor (log e base two). The accuracy of the reduced argument thus depends on the size of N.

MULF
MULD

Multiply Floating/Double 171(AC)FSRC



Format: MULF FSRC,AC

Operation: Let PROD = AC * (fsrc).

If underflow occurs and FIU is not enabled, AC ◂— exact 0.

If overflow occurs and FIV is not enabled, AC ◂— exact 0.

For all other cases, AC ◂— PROD.

Condition Codes: FC ← 0
FV ← 1 if overflow occurs, else FV ← 0
FZ ← 1 if AC = 0, else FZ ← 0
FN ← 1 if AC < 0, else FN ← 0

Description: If the biased exponent of either operand is zero, (AC) ← exact 0. For all other cases, PROD is generated to 48 (32 for KEF11-AA) bits for floating mode and 59 (64 for KEF11-AA) bits for double mode. The product is rounded or chopped according to the value of the FT bit, and is stored in AC except for:

1. Overflow with interrupt disabled
2. Underflow with interrupt disabled

For these exceptional cases, an exact zero is stored in AC.

Interrupts: If FIUV is enabled, trap on "−0" in (fsrc) occurs before execution.

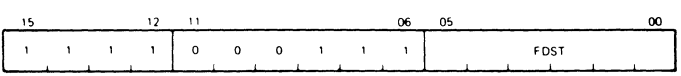
If overflow or underflow occurs and if the corresponding interrupt is enabled, the trap occurs with the faulty result in AC. The fractional parts are correctly stored. The exponent part is too small by 400_8 for overflow. It is too large by 400_8 for underflow, except for the special case of zero, which is correct.

Accuracy: Errors due to overflow and underflow are described above. If neither occurs, the error incurred is bounded by one LSB in chopping mode and ½ LSB in rounding mode.

Special Comment: The undefined variable "−0" can occur only in conjunction with overflow or underflow. It will be stored in AC, only if the corresponding interrupt is enabled.

NEGF
NEGD

Negate Floating/Double 1707 FDST



Format: NEGF −(fdst)

Operation: (fdst) ◂— −(fdst) if EXP [(fdst)] ≠ 0, else (fdst) ◂— exact 0.

Condition Codes:
FC ◂— 0
FV ◂— 0
FZ ◂— 1 if (fdst) = 0, else FZ ◂— 0
FN ◂— 1 if (fdst) < 0, else FN ◂— 0

Description: Negate single-precision or double-precision number, store result in same location (fdst).

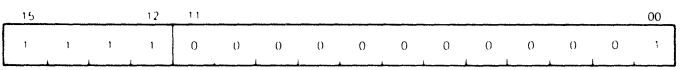
Interrupts: If FIUV is enabled, trap on −0 occurs after execution.

Overflow and underflow cannot occur.

Accuracy: These instructions are exact.

Special Comment: If a −0 is present in memory and the FIUV bit is enabled, then the floating-point processor stores an exact zero in memory. If a negative number is present, then the floating-point processor stores the actual negative result in memory. The condition codes reflect an exact zero (FZ ◂— 1).

SETF
Set Floating Mode 170001

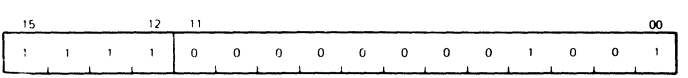


Format: SETF

Operation: FD ◂— 0

Description: Set the floating-point option in single-precision mode.

SETD
Set Floating Double Mode 170011



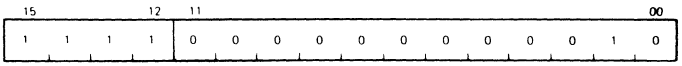
Format: SETD

Operation: FD ← 1

Description: Set the floating-point option in double-precision mode.

SETI

Set Integer Mode 177002



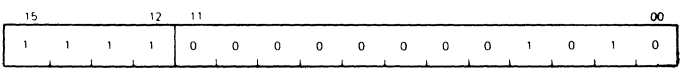
Format: SETI

Operation: FL ← 0

Description: Set the floating-point option for short-integer data.

SETL

Set Long-Integer Mode 177012



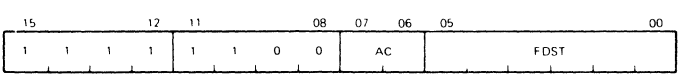
Format: SETL

Operation: FL ← 1

Description: Set the floating-point option for long-integer data.

STCFD
STCDF

Store and Convert from Floating to Double and from Double to Floating 176(AC)FDST



Format: STCFD AC,FDST

Operation: If AC = 0, (fdst) ◂— exact 0.

If FD = 1, FT = 0, FIV = 0 and rounding causes overflow, (fdst) ◂— exact 0.

In all other cases, (fdst) ◂— Cxy[AC], where Cxy specifies conversion from floating mode x to floating mode y.

x = F, y = D if FD = 0 (single) STCFD
x = D, y = F if FD = 1 (double) STCDF

Condition Codes: FC ◂— 0
FV ◂— 1 if conversion produces overflow, else FV ◂— 0
FZ ◂— 1 if AC = 0, else FZ ◂— 0
FN ◂— 1 if AC < 0, else FN ◂— 0

Description: If the current mode is single-precision, the accumulator is stored left-justified in FDST and the lower half is cleared.

If the current mode is double-precision, the contents of the accumulator are converted to single-precision, chopped, or rounded, depending on the state of FT, and then stored in FDST.

Interrupts: Trap on −0 will not occur, even if FIUV is enabled, because FSRC is an accumulator.

Underflow cannot occur.

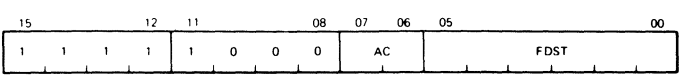
Overflow cannot occur for STCFD.

A trap occurs if FIV is enabled, and if rounding with STCDF causes overflow. (fdst) ◂— overflowed result. This must be +0 or −0.

Accuracy: STCFD is an exact instruction. Except for overflow, described above, STCDF incurs an error bounded by 1 LSB in chopping mode and by ½ LSB in rounding mode.

STF
STD

Store Floating/Double 174(AC)FDST



Format: STF AC,FDST

Operation: (fdst) ◂— AC

Condition Codes: FC ◂— FC
FV ◂— FV
FZ ◂— FZ
FN ◂— FN

Description: Store single-precision or double-precision number from AC.

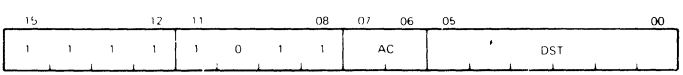
Interrupts: These instructions do not interrupt if FIUV is enabled, because the −0, if present, is in AC, not in memory.

Overflow and underflow cannot occur.

Accuracy: These instructions are exact.

Special Comment: These instructions permit storage of a −0 in memory from AC. There are two conditions in which −0 can be stored in AC of the floating-point processor. One occurs when underflow or overflow is present and the corresponding interrupt is enabled. A second occurs when an LDF, LDD, LDCDF, or LDCFD instruction is executed and the FIUV bit is disabled.

STCFI STCDI
STCFL STCDL
Store and Convert from Floating or Double to Integer or Long Integer 175(AC + 4)DST



Format: STCFI AC,DST

Operation: (dst) ◂— Cxj[AC] if −JL − 1 < Cxj[AC] < JL + 1, else (dst) ◂— 0, where Cjx specifies conversion from floating mode x to integer mode j.

j = I if FL = 0, j = L if FL = 1
x = F if FD = 0, x = D if FD = 1

JL is the largest integer

$2^{15} - 1$ for FL = 0
$2^{32} - 1$ for FL = 1

Condition Codes: C, FC ◂— 0 if $-JL-1 < Cxj[AC] < JL+1$,
else C, FC ◂— 1
V, FV ◂— 0
Z, FZ ◂— 1 if (dst) = 0, else Z, FZ ◂— 0
N, FN ◂— 1 if (dst) < 0, else N, FN ◂— 0

Description: Conversion is performed from a floating-point representation of the data in the accumulator to an integer representation.

If the conversion is to a 32-bit word (L mode) and an addressing mode of 0 or immediate addressing mode is specified, only the most significant 16 bits are stored in the destination register.

If the operation is out of the integer range selected by FL, FC is set to 1 and the contents of the dst are set to 0.

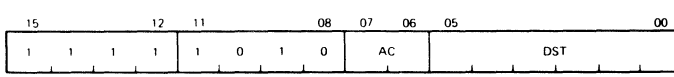
Numbers to be converted are always chopped (rather than rounded) before conversion. This is true even when the chop mode bit FT is cleared in the FPS register.

Interrupts: These instructions do not interrupt if FIUV is enabled, because the −0, if present, is in AC, not in memory.

If FIC is enabled, trap on conversion failure will occur.

Special Comment: These instructions store the integer part of the floating-point operand, which may not be the integer most closely approximating the operand. They are exact if the integer part is within the range implied by FL.

STEXP
Store Exponent 175(AC)DST



Format: STEXP AC,DST

Operation: (dst) ◂— EXP[AC] – 200_8

Condition Codes: C, FC ◂— 0
V, FV ◂— 0
Z, FZ ◂— 1 if (dst) = 0, else Z, FZ ◂— 0
N, FN ◂— 1 if (dst) < 0, else N, FN ◂— 0

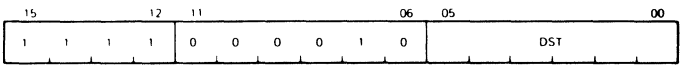
Description: Convert AC's exponent from excess 200_8 notation to 2's complement and store the result in dst.

Interrupts: This instruction will not trap on –0.

Overflow and underflow cannot occur.

Accuracy: This instruction is always exact.

STFPS
Store Floating-point Processor's Program Status 1702 DST



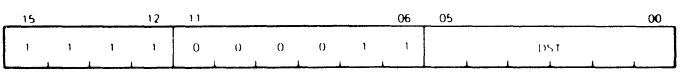
Format: STFPS DST

Operation: (dst) ◂— FPS

Description: Store floating-point status register in dst.

Special Comment: Bits 13, 12, and 4 (if maintenance mode is not implemented) are stored as 0. All other bits are the corresponding bits in the FPS.

STST
Store Floating-point Processor's Status 1703 DST



Format: STST DST

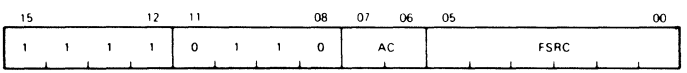
Operation: (dst) ◂— FEC
(dst + 2) ◂— FEA

Description: Store the FEC and FEA in dst and dst + 2.

NOTE
1. If the destination mode specifies a general register or immediate addressing, only the FEC is saved.
2. The information in these registers is current only if the most recently executed floating-point instruction caused a floating-point exception.

SUBF
SUBD

Subtract Floating/Double 173(AC)FSRC



Format: SUBF FSRC,AC

Operation: Let DIFF = AC − (fsrc).

If underflow occurs and FIU is not enabled, AC ◂— exact 0.

If overflow occurs and FIV is not enabled, AC ◂— exact 0.

For all cases, AC ◂— DIFF.

Condition Codes: FC ◂— 0
FV ◂— 1 if overflow occurs, else FV ◂— 0
FZ ◂— 1 if AC = 0, else FZ ◂— 0
FN ◂— 1 if AC < 0, else FN ◂— 0

Description: Subtract the contents of fsrc from the contents of AC. The subtraction is carried out in single-precision or double-precision and is rounded or chopped according to the values of the FD and FT bits in the FPS register. The result is stored in AC except for:

1. Overflow with interrupt disabled.
2. Underflow with interrupt disabled.

For these exceptional cases, an exact zero is stored in AC.

Interrupts: If FIUV is enabled, trap on −0 in fsrc occurs before execution.

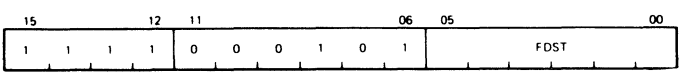
If overflow or underflow occurs and if the corresponding interrupt is enabled, the trap occurs with the faulty result in AC. The fractional parts are correctly stored. The exponent part is too small by 400_8 for overflow. It is too large by 400_8 for underflow, except for the special case of zero, which is correct.

Accuracy: Errors due to overflow and underflow are described above. If neither occurs, then for like-signed operands with an exponent difference of zero or one, the answer returned is exact, if a loss of significance of one or more bits can occur. Note that these are the only cases for which loss of significance of more than one bit can occur. For all other cases, the result is inexact with error bounds of:

1. 1 LSB in truncated mode with either single-precision or double-precision
2. From ½ LSB to ¾ LSB in rounding mode depending on floating-point processor. See Appendix B—PDP-11 Family Differences—for details.

Special Comment: The undefined variable −0 can occur only in conjunction with overflow or underflow. It will be stored in AC only if the corresponding interrupt is enabled.

TSTF
TSTD
Test Floating/Double 1705 FDST



Format: TSTF FDST

Operation: (fdst)

Condition Codes:
FC ◂— 0
FV ◂— 0
FZ ◂— 1 if (fdst) = 0, else FZ ◂— 0
FN ◂— 1 if (fdst) < 0, else FN ◂— 0

Description: Set the FP11 condition codes according to the contents of fdst.

Interrupts: If FIUV is set, trap on −0 occurs after execution.

Overflow and underflow cannot occur.

Accuracy: These instructions are exact.

SECTION C: CHARACTER AND DECIMAL STRING INSTRUCTIONS (CIS)

Operands for these instructions are identified by *descriptors*. The descriptors are located as follows:

- *Register* form instructions (mnemonic does not end in "I"). Descriptors are located in general registers.
- *In-line* form instructions (mnemonic ends in "I"). Pointers to descriptors follow the instruction.

When the description of an instruction describes the result as "unpredictable," it means that the effect of the instruction is indeterminate and nonrepeatable.

ADDN **ADd Decimal Numeric** **076050_8**

ADDP **ADd Decimal Packed** **076070_8**

Operands: R0,R1: Source 1 Descriptor
R2,R3: Source 2 Descriptor
R4,R5: Destination Descriptor

ADDNI **ADd Decimal Numeric (In line)** **076150_8**

ADDPI **ADd Decimal Packed (In line)** **076170_8**

Operands: addr + 2: pointer to Source 1 Descriptor
addr + 4: pointer to Source 2 Descriptor
addr + 6: pointer to Destination Descriptor

Operation: (dst) ← (src 1) + (src 2)
(R0),(R1),(R2),(R3) ← 0 (ADDN, ADDP only)

Condition Codes: N: Set if (dst) < 0
Z: Set if (dst) = 0
V: Set if dst cannot contain all significant digits of the result
C: cleared

Description: Source string 1 is added to source string 2 and the result is stored in the destination string. When the instruction is completed, the source descriptor registers (R0–R3) are cleared (ADDN, ADDP only).

Notes:

1. The operation of these instructions is unaffected by any overlap of the source strings provided that each source string is a valid representation of the specified data type.
2. Source strings may overlap the destination string only if all corresponding digits of the strings are in coincident bytes in memory.

ASHN **Arithmetic SHift Numeric** **076056_8**

ASHP **Arithmetic SHift Packed** **076076_8**

Operands: R0,R1: Source Descriptor
R2,R3: Destination Descriptor
R4:Shift Descriptor:
R4 <7:0>: Shift Count
R4 <11:8>: Rounding Digit
R4 <15:12>: 0

ASHNI **Arithmetic SHift Numeric (In-line)** **076156_8**

ASHPI **Arithmetic SHift Packed (In-line)** **076176_8**

Operands: addr + 2: pointer to Source Descriptor
addr + 4: pointer to Destination Descriptor
addr + 6: Shift Descriptor
addr + 6 <7:0>: Shift Count
addr + 6 <11:8>: Rounding Digit
addr + 6 <15:12>: 0

Operations: (dst) ← (src) × (10 ** shift count)
(R0),(R1),(R4) ← 0 (ASHN, ASHP only)

Condition Codes: N: Set if (dst) < 0
Z: Set if (dst) = 0
V: Set if dst cannot contain all significant digits of the result
C: Cleared

Description: The decimal number specified by the source descriptor is arithmetically shifted and stored in the area specified by the destination descriptor. The shifted result is aligned with the least significant digit position in the destination string. The shift count is a

2's complement byte whose value ranges from -128_{10} to $+127_{10}$. If the shift count is positive, a shift in the direction of least to most significant digits is performed. A negative shift count performs a shift from most to least significant digit. Thus, the shift count is the power of ten by which the course is multiplied; negative powers of ten effectively divide. Zero digits are supplied for vacated digit positions. A zero shift count will move the source to the destination. The condition codes reflect the value stored in the destination string, and whether all significant digits were stored.

A negative shift count invokes a rounding operation. The result is constructed by shifting the source the specified number of digit positions. The rounding digit is then added to the most significant digit which was shifted out. If this sum is less than 10_{10} the shifted result is stored in the destination string. If the sum is 10_{10} or greater, the magnitude of the shifted result is increased by 1 and then stored in the destination string. If no rounding is desired, the rounding digit should be zero.

The shift count and rounding digit are represented in a single word referred to as the shift descriptor. Bits <15:12> of this word must be zero.

Notes:

1. If bits <15:12> of the shift descriptor word are not zero, the effect of the instruction is unpredictable.
2. If bits <11:8> of the shift descriptor are not a valid decimal digit, the results of the instruction are unpredictable.
3. Any overlap of the source and destination strings will produce unpredictable results.

The source descriptor registers (R0, R1) and the shift word register (R4) are cleared when the instruction is completed (ASHN, ASHP only).

CMPC **CoMPare Character** 076044_8

Operands: R0,R1: Source 1 Descriptor
R2,R3: Source 2 Descriptor
R4: Fill Character
R4<7:0>: Fill Character
R4<15:8>: 0

CMPCI **CoMPare Character (In-line)** 076144_8

Operands: addr + 2: pointer to Source 1 Descriptor
addr + 4: pointer to Source 2 Descriptor

Operation: Source string 1 is compared to Source string 2 (src1 – src2).

Condition Codes: The condition codes are based on the arithmetic comparison of the most significant pair of unequal src1 and src2 characters (src1.byte-src2.byte).
N: Set if result < 0; cleared otherwise
Z: Set if result = 0; cleared otherwise
V: Set if there was arithmetic overflow, that is, src1.byte <7> and src2.byte <7> were different, and src2.byte <7> was the same as bit <7> of src1.byte – src2.byte; cleared otherwise
C: cleared if there was a carry from the most significant bit of the result; set otherwise

Description: Each character of src1 is compared with the corresponding character of src2 by examining the character strings from most significant to least significant characters. If the character strings are of unequal length, the shorter character string is conceptually extended to the length of the longer character string with fill characters beyond its least significant character. The instruction terminates when the first corresponding unequal characters are found or when both character strings are exhausted. The condition codes reflect the last comparison, permitting the unsigned branch instructions to test the result.

CMPC only: The instruction terminates with substring descriptors in R0,R1 and R2,R3 which represent the portion of each source character string beginning with the most significant corresponding unequal characters. R0,R1 contain a descriptor for the

unequal position of the original src1 string; R2,R3 contain a descriptor for the unequal position of the original src2 string. A vacant character-string descriptor indicates that the entire source character string was equal to the corresponding position of the other source character string, including extension by the fill character; its address is one greater than that of the least significant character of the character string.

Notes:

1. The operation of this instruction is unaffected by any overlap of the source character strings.
2. If the src1 character string is vacant, the fill character will be compared with src2. If the src2 character string is vacant, the fill character will be compared with src1. If both character strings are vacant, the condition codes will indicate equality.
3. CMPC. If an initial source character-string descriptor is vacant, the resulting substring descriptor is the same as the original character-string descriptor.
4. A test for success is BEQ; a test for failure is BNE.
5. When the instruction terminates, the condition codes will be set as if a CMPB instruction operated on the most significant unequal characters. If both strings are initially vacant or are identical, the condition codes will be set as if the last characters to be compared were identical. This results in equality with N cleared, Z set, V cleared, and C cleared.
6. Both CMPC and CMPCI update the condition codes. CMPC returns substring descriptors.

CMPN	**CoMPare Numeric**	076052_8
CMPP	**CoMPare Packed**	076072_8

Operands: R0,R1: Source 1 Descriptor
R2,R3: Source 2 Descriptor

CMPNI	**CoMPare Numeric (In-line)**	076152_8
CMPPI	**CoMPare Packed (In-line)**	076172_8

Operands: addr + 2: pointer to Source 1 Descriptor
addr + 4: pointer to Source 2 Descriptor

Operation: Source string 1 is compared to
Source string 2 (src1 - src2)
(R0),(R1),(R2),(R3) ← 0 (CMPN, CMPP only)

Condition Codes: N: Set if src1 < src2; cleared otherwise
Z: Set if src1 = src2; cleared otherwise
V: Cleared
C: Cleared

Description: Src1 is arithmetically compared with src2. The condition codes reflect the comparison. The signed branch instruction can be used to test the result.

When the instruction is completed, the source descriptor registers (R0–R3) are cleared (CMPN, CMPP only).

Note: The operation of these instructions is unaffected by any overlap of the source strings, provided that each source string is a valid representation of the specified data type.

CVTLN **ConVerT Long to decimal Numeric** **076057_8**

CVTLP **ConVerT Long to decimal Packed** **076077_8**

Operands: R0, R1: Destination Descriptor
R2, R3: Source Long Integer
R2<15>: Sign
R2<14:0>: High-Order
R3<15:0>: Low-Order

CVTLNI **ConVerT Long to decimal Number (In-line)** **076157_8**

CVTLPI **ConVerT Long to decimal Packed (In-line)** **076177_8**

Operands: addr + 2 : pointer to Destination Descriptor
addr + 4 : pointer to Source Long Integer
pointer <15:0>: Low-Order
pointer +2<15>: Sign
pointer +2<14:0>: High-Order

Operation: Decimal string ← long integer
(R2),(R3) ← 0 (CVTLN, CVTLP only)

Condition Codes:
N: Set if dst < 0; cleared otherwise
Z: Set if dst = 0; cleared otherwise
V: Set if dst cannot contain all significant digits of the result; cleared otherwise
C: Cleared

Description: The source long integer is converted to a decimal string. The condition codes reflect the result stored in the destination decimal string and whether all significant digits were stored. When the instruction is completed, the source long integer registers (R2, R3) are cleared (CVTLN, CVTLP only).

Notes:

1. CVTLN and CVTLP use a long integer with the sign and high-order portion in R2, and the low-order portion in R3.
2. CVTLNI and CVTLPI use a long integer with the sign and high-order portion at the pointer address +2, and the low-order portion at the pointer address.

CVTNL **ConVerT decimal Numeric to Long** **076053_8**

CVTPL **ConVerT decimal Packed to Long** **076073_8**

Operands:
R0,R1: Source Descriptor
R2,R3: Destination Long Integer
R2<15>: Sign
R2<14:0>: High-Order
R2<13:0>: Low-Order

CVTNLI **ConVerT decimal Numeric to Long (In-line)** **076153_8**

CVTPLI **ConVerT decimal Packed to Long (In-line)** **076173_8**

Operands:
addr + 2 : pointer to Source Descriptor
addr + 4 : pointer to Destination Long Integer
pointer<15:0>: Low-Order

pointer + 2<15>: Sign
pointer +2<14:0>: High-Order

Operation: Long integer ← decimal string
(R0),(R1) ← 0 (CVTNL, CVTPL only)

Condition Codes: The condition codes are based on the long integer destination and on the sign of the source decimal string.

N: Set if long integer < 0; cleared otherwise
Z: Set if long integer = 0; cleared otherwise
V: Set if long integer dst cannot correctly represent the 2's complement form of the result; cleared otherwise
C: Set if src < 0 and long.integer ≠ 0; cleared otherwise

Description: The source decimal string is converted to a long integer. The condition codes reflect the result of the operation, and whether significant digits were not converted.

When the instruction is completed, the source descriptor registers (R0, R1) are cleared (CVTNL, CVTPL only).

Notes:

1. CVTNL and CVTPL use a long integer with the sign and high-order portion in R2 and the low-order portion in R3.
2. CVTNLI and CVTNPI use a long integer with the sign and high-order portion at the pointer address +2 and the low-order portion at the pointer address.
3. If the V bit is set, the contents of the long integer designation are the least significant 32 bits of the result.
4. A source whose value is $+2^{31}$ can be represented as a 32-bit binary integer. However, since the destination is a 2's complement long integer, the resulting condition codes will be N set, Z cleared, V set, and C cleared.

CVTNP **ConVerT Numeric to Packed** 076055_8

CVTPN **ConVerT Packed to Numeric** 076054_8

Operands: R0,R1: Source Descriptor
R2,R3: Destination Descriptor

CVTNPI **ConVerT Numeric to Packed (In-line)** 076155_8

CVTPNI **ConVerT Packed to Numeric (In-line)** 076154_8

Operands: addr + 2: pointer to Source Descriptor
addr + 4: pointer to Destination Descriptor

Operation: Packed string ← numeric string (CVTNP, CVTNPI)
Numeric string ← packed string (CVTPN, CVTPNI)
(R0),(R1) ← 0 (CVTNP, CVTPN only)

Condition Codes:
N: Set if dst < 0; cleared otherwise
Z: Set if dst = 0; cleared otherwise
V: Set if dst cannot contain all significant digits of the result; cleared otherwise
C: Cleared

Description: These instructions convert between numeric and packed decimal strings. The source decimal string is converted and moved to the destination string. The condition codes reflect the result of the operation and whether all significant digits were stored.

When the instruction is completed, the source descriptor registers (R0, R1) are cleared (CVTNP, CVTPN only).

Notes:

1. The results of the instruction are unpredictable if the source and destination strings overlap.
2. These instructions use both a numeric and a packed decimal-string descriptor.

DIVP **DIVide Packed** 076075_8

Operands: R0,R1: Source 1 Descriptor
R2,R3: Source 2 Descriptor
R4,R5: Destination Descriptor

DIVPI **DIVide Packed (In-line)** **076175_8**

Operands: addr + 2: pointer to Source 1 Descriptor
addr + 4: pointer to Source 2 Descriptor
addr + 6: pointer to Destination Descriptor

Operation: (dst) ← (src2/src1)
(R0),(R1),(R2),(R3) ← 0 (DIVP only)

Condition Codes:
N: Set if dst < 0; cleared otherwise
Z: Set if dst = 0; cleared otherwise
V: Set if dst cannot contain all significant digits of the result or if src1 = 0; cleared otherwise
C: Set if src1 = 0; cleared otherwise

Description: Src2 is divided by src1, and the quotient (fraction truncated) is stored in the destination string. The condition codes reflect the value stored in the destination string and whether all significant digits were stored.

When the instruction is completed, the source descriptor registers (R0–R3) are cleared (DIVP only).

Notes:

1. The operation of these instructions is unaffected by any overlap of the source strings provided that each source string is a valid representation of the specified data type.
2. The results of the instruction are *unpredictable* if the source and destination strings overlap.
3. Division by zero will set the V and C bits. The destination string and the N and Z condition code bits will be *unpredictable.*
4. No numeric string divide instruction is provided.

LOCC **LOCate Character** **076040_8**

Operands: R0, R1: Source Descriptor
R4: Search Character
R4<7:0>: Character
R4<15:8>: 0

LOCCI **LOCate Character (In-line)** **076140_8**

Operands:
addr + 2: pointer to Source Descriptor
addr + 4: Search Character
addr + 4<7:0>: Character
addr + 4<15:8>: 0
R0,R1: returned Character String Descriptor

Operation:
Search source character string for a specified character.

Condition Codes:
The condition codes are based on the final contents of R0.
N: Set if R0 <15> set; cleared otherwise
Z: Set if R0 = 0; cleared otherwise
V: Cleared
C: Cleared

Description:
The source character string is searched from most significant to least significant character until the first occurrence of the search character. A character-string descriptor is returned in R0–R1 which represents the portion of the source character string beginning with the located character. If the source character string contains only characters not equal to the search character, the instructions return a vacant character-string descriptor with an address one greater than that of the least significant character of the source character string. The condition codes reflect the resulting value in R0.

Notes:

1. If the initial source character-string descriptor is vacant, the instruction terminates with the condition codes indicating no match was found. The original source character-string descriptor is returned in R0,R1.
2. A test for success is BNE; a test for failure is BEQ.
3. The condition codes will be set as if this instruction were followed by TST R0.

L2DR **Load 2 DescriptoRs** **$07606r_8$**

Operands:
(Rr): address of pointer to Alpha Descriptor
(Rr) + 2: address of pointer to Beta Descriptor

Operation: (R0,R1) ← Alpha Descriptor
(R2,R3) ← Beta Descriptor

Condition Codes: Unaffected

Description: This instruction augments the character and decimal-string instructions by efficiently loading string descriptors into the general registers.

The descriptor "Alpha" is loaded into R0,R1; a second descriptor "Beta" is loaded into R2–R3. The address of the descriptors is determined by the addressing mode @(Rr)+ where r is the low-order three bits of the opcode word. The address of the descriptor "Alpha" is derived by applying this addressing mode once; the address of the descriptor "Beta"" is derived by applying this addressing mode a second time. The addressing mode autoincrements the indicated register by two. The addressing mode computation is not affected by the descriptors which are loaded into the general registers. The words that contain the addresses of the descriptors are in consecutive words in memory; the descriptions themselves may be anywhere in memory. The condition codes are not affected.

When the instruction is completed, the "Alpha" descriptor is in R0,R1 and the "Beta" descriptor is in R2,R3.

L3DR **Load 3 DescriptoRs** **$07606r_8$**

Operands: (Rr): address of pointer to Alpha Descriptor
(Rr) + 2: address of pointer to Beta Descriptor
(Rr) + 4: address of pointer to Gamma Descriptor

Operation: (R0,R1) ← Alpha Descriptor
(R2,R3) ← Beta Descriptor
(R4,R5) ← Gamma Descriptor

Condition Codes: Unaffected

Description: Same as L2DR, except the address of the descriptor "Gamma" is derived by applying addressing mode @(Rr)+ a third time, and the Gamma Descriptor is loaded into R4,R5.

MATC **MATch Character** **076045_8**

Operands: R0,R1: Source Descriptor
R2,R3: Object Descriptor

MATCI **MATch Character (In-line)** **076145_8**

Operands: addr + 2: pointer to Source Descriptor
addr + 4: pointer to Object Descriptor
R0,R1: result of substring descriptor

Operation: Search source character string for object character string.

Condition Codes: The condition codes are based on the final contents of R0.
N: Set if R0 <15> set; cleared otherwise
Z: Set if R0 = 0; cleared otherwise
V: Cleared
C: Cleared

Description: The source character string is searched from most significant to least significant character for the first occurrence of the entire object character string. A character-string descriptor is returned in R0–R1 which represents the portion of the original source character string from the most significant character which completely matches the object character string to the end of the source character string. If the object character string did not completely match any portion of the source character string, the character descriptor returned in R0–R1 is vacant with an address one greater than the least significant character in the source string. The condition codes reflect the resulting value in R0. If the Z bit is cleared, the entire object was successfully matched with the source character string; if the Z bit is set, the match failed.

Notes:

1. The operation of this instruction is unaffected by any overlap of the source and object character strings.
2. A vacant object character string matches any nonvacant source character string. A vacant source character string will not match any object character string. If the initial source character string descriptor is vacant, the instruction terminates with the condition codes indicating no match was found. The original source character string descriptor is returned in R0,R1.
3. If the length of the object character string is greater than that of the source character string, no match is found; R0,R1 and the condition codes will be updated.
4. A test for success is BNE; a test for failure is BEQ.
5. The condition codes will be set as if this instruction were followed by TST R0.

MOVC **MOVe Character** 076030_8

MOVRC **MOVe Reverse-justified Character** 076031_8

Operands:
R0,R1: Source Descriptor
R2,R3: Destination Descriptor
R4: Fill Character
R4<7:0>: Character
R4(15:0>: 0

MOVCI **MOVe Character (In-line)** 076130_8

MOVRCI **MOVe Reverse-justified Character (In-line)** 076131_8

Operands:
addr + 2: pointer to Source Descriptor
addr + 4: pointer to Destination Descriptor
addr + 6: Fill character
addr + 6<7:0>: Character
addr + 6<15:8>: 0

Operation:

(dst) ← (src)(MOVC, MOVCI)
(dst) ← reverse-justified (src)(MOVRC,MOVRCI)
(R0) ← max (0, source length–dest length)
(MOVC,MOVRC only)
(R1),(R2),(R3) ← 0 (MOVC, MOVRC only)

Condition Codes:

The condition codes are based on the arithmetic comparison of the initial character string lengths (result = src.len-dst.len).

N: Set if result < 0; cleared otherwise
Z: Set if result = 0; cleared otherwise
V: Set if there was arithmetic overflow, that is, src.len <15> and des.len <15> were different, and dst.len <15> was the same bit <15> of (src.len-dst.len); cleared otherwise
C: Cleared if there was a carry from the most significant bit of the result; set otherwise

Description:

The character string specified by the source descriptor is moved into the area specified by the destination descriptor.

MOVC, MOVCI: The string is aligned by most significant character.
MOVCR, MOVCRI: The string is aligned by least significant character.

The condition codes reflect an arithmetic comparison of the original source and destination lengths. If the source string is shorter than the destination string, the fill character is used to complete the most significant part of the destination string. This is indicated by the C bit set. If the source string is longer than the destination string, the most significant characters of the source string are not moved. This is indicated by the Z and C bits cleared. If the source and destination strings are of equal length, all characters are moved with neither truncation nor filling. This is indicated by the Z bit set. The unsigned branch instructions may test the result of the instruction.

When the instruction is completed, R0 contains the number of unmoved source string characters, and R1 through R3 are cleared (MOVC, MOVRC only).

Notes:

1. The operation of this instruction is unaffected by any overlap of the source and destination strings. The result is equivalent to having read the entire source string before storing characters in the destination.
2. If the source string is vacant, the fill character will be propagated through the destination string. If the destination string is vacant, no characters will be moved. Condition codes will be updated. MOVC, MOVRC will update the general registers.
3. MOVC, MOVRC. When the instruction terminates, R0 is zero only if Z or C are set.
4. The condition codes will be set as if this instruction were preceded by CMP src.len, dst.len.

MOVTC **MOVe Translated Character** **076032_8**

Operands: R0,R1: Source Descriptor
R2,R3: Destination Descriptor
R4: Fill Character
R4(7:0>: Character
R4<15:8>: 0
R5: pointer to 256-byte table of translation characters

MOVTCI **MOVe Translated Character (In-line)** **076132_8**

Operands: addr + 2: pointer to Source Descriptor
addr + 4: pointer to Destination Descriptor
addr + 6: Fill Character
addr + 6<7:0>: Character
addr + 6<15:8>: 0
addr + 8: pointer to 256-byte table of translation characters

Operation: (dst) ← translated (src)
(R0) ← max (0, source length-dest length) (MOVTC only)
(R1), (R2), (R3) ← 0 (MOVTC only)

Condition Codes:

The condition codes are based on the arithmetic comparison of the initial character string lengths (result = src.len-dst.len).
N: Set if result < 0; cleared otherwise
Z: Set if result = 0; cleared otherwise
V: Set if there was arithmetic overflow, that is, src.len <15> and dst.len <15> were different, and dst.len <15> was the same as bit <15> of (src.len-dst.len); cleared otherwise
C: Cleared if there was a carry from the most significant bit of the result; set otherwise

Description:

The character string specified by the source descriptor is translated and moved into the area specified by the destination descriptor. It is aligned by the most significant character. Translation is accomplished by using each source character as an 8-bit positive integer index into a 256-byte table, the address of which is an operand of the instruction. The byte at the indexed location in the table is stored in the destination string. The condition codes reflect an arithmetic comparison of the original source and destination.

If the source string is shorter than the destination string, the untranslated fill character is used to complete the least significant part of the destination string. This is indicated by the C bit set. If the source string is longer than the destination string, the least significant characters of the source string are not moved. This is indicated by the Z and C bits cleared. If the source and destination strings are of equal length, all characters are translated and moved with neither truncation nor filling. This is indicated by the Z bit set. The unsigned branch instructions may test the result of the instruction.

When the instruction is completed, R0 contains the number of unmoved source string characters, and R1 through R3 are cleared.

Notes:

1. The operation of this instruction is unaffected by any overlap of the source and destination strings. The result is equivalent to having read the entire source string before storing characters in the destination.

2. If the destination string overlaps the translation table in any way, the results of the instruction will be *unpredictable.*
3. If the source string is vacant, the untranslated fill character will be propagated through the destination string. If the destination string is vacant, no characters will be moved. Condition codes will be updated. MOVTC will update the general registers.
4. MOVTC. When the instruction terminates, R0 is zero only if Z or C are set.
5. The condition codes will be set as if this instruction were preceded by CMP src.len, dst.len.
6. The effect of the instruction is *unpredictable* if the entire 256-byte translation table is not readable memory.

MULP **MULtiply Packed** **076074_8**

Operands: R0,R1: Source 1 Descriptor
R2,R3: Source 2 Descriptor
R4,R5: Destination Descriptor

MULPI **MULtiply Packed (In-line)** **076174_8**

Operands: addr + 2: pointer to Source 1 Descriptor
addr + 4: pointer to Source 2 Descriptor
addr + 6: pointer to Destination Descriptor

Operation: (dst) ←(src1) × (src2)
(R0),(R1),(R3) ← 0 (MULP only)

Condition Codes: N: Set if (dst) < 0
Z: Set if (dst) = 0
V: Set if dst cannot contain all significant digits of the result
C: Cleared

Description: Src1 and src2 are multiplied, and the result is stored in the destination string. The condition codes reflect the value stored in the destination string and whether all significant digits were stored.

When the instruction is completed, the source descriptor registers (R0,R3) are cleared (MULP only).

Notes:

1. The operation of MULP, MULPI is unaffected by any overlap of the source strings provided that each source string is a valid representation of the specified data type.
2. The results are *unpredictable* if the source and destination strings overlap.
3. No numeric string multiply instruction is provided.

SCANC **SCAN Character** **076042_8**

Operands: R0,R1: Source Descriptor
R4,R5: Character Set Descriptor

SCANCI **SCAN Character (In-line)** **076142_8**

Operands: addr + 2: pointer to Source Descriptor
addr + 4: pointer to Character Set Descriptor
R0,R1: result substring descriptor

Operation: Search source character string for a member of the character set.

Condition Codes: The condition codes are based on the final contents of R0.
N: Set if R0 <15> set; cleared otherwise
Z: Set if R0 = 0; cleared otherwise
V: Cleared
C: Cleared

Description: The source character string is searched from most significant to least significant character until the first occurrence of a character which is a member of the character set. A character-string descriptor is returned in R0,R1 which represents the portion of the source character string beginning with the located member of the character set. If the source character string contains only characters which are not in the character set, the instructions return a vacant character-string descriptor with an address one greater than that of the least significant character of the source character string. The condition codes reflect the resulting value in R0.

When the instruction is completed, R0,R1 contain a character-string descriptor which represents the substring of the source character string beginning with the most significant character which is a member of the character set.

Notes:

1. If the initial source character-string descriptor is vacant, the instruction terminates with the condition codes indicating that no characters in the set were found. The original source character-string descriptor is returned in R0-R1.
2. The source character string and character set table may overlap in any way.
3. A test for success is BNE; a test for failure is BEQ.
4. The condition codes will be set as if this instruction were followed by TST R0.
5. The effect of the instruction in *unpredictable* if the entire 256-byte character set table is not in readable memory.

SKPC **SKiP Character** **076041_8**

Operands: R0–R1: Source Descriptor
R4: Search Character
R4<7:0>: Character
R4<15:8>: 0

SKPCI **SKiP Character (In-line)** **076141_8**

Operands: addr + 2: pointer to Source Descriptor
addr + 4: Search Character
addr + 4<7:0>: Character
addr + 4<15:7>: 0
R0,R1: result substring character descriptor

Operation: Search the source character string until a character other than the search character is found.

Condition Codes: The condition codes are based on the final contents of R0.

N: Set if R0 <15> set; cleared otherwise
Z: Set if R0 = 0; cleared otherwise
V: Cleared
C: Cleared

Description: The source character string is searched from most significant to least significant character until the first occurrence of a character which is not the search character. A character-string descriptor is returned in R0,R1 which represents the portion of the source character string beginning the most significant character which was not equal to the search character. If the source character string contains only characters equal to the search character, the instruction returns a vacant character-string descriptor with an address one greater than that of the least significant character of the source character string. The condition codes reflect the resulting value in R0.

When the instruction is completed, R0,R1 contain a character-string descriptor which represents the substring of the source character string beginning with the most significant character which was not equal to the search character.

Notes:

1. If the initial source character-string descriptor is vacant, the instruction terminates with the condition codes indicating the character string only contained search characters. The original source character string descriptor is returned in R0,R1.
2. The condition codes will be set as if this instruction were followed by TST R0.

SPANC **SPAN Character** **076042_8**

Operands: R0,R1: Source Descriptor
R4,R5: Character Set Descriptor

SPANCI **SPAN Character (In-line)** **076143_8**

Operands: addr + 2: pointer to Source Descriptor
addr + 4: pointer to Destination Descriptor
R0,R1: result character substring descriptor

Operation: Search same character string for a character which is not a member of the character set.

Condition Codes: Search source character string for a character which is not a member of the character set.

The condition codes are based on the final contents of R0.
N: Set if R0<15> set; cleared otherwise
Z: Set if R0 = 0; cleared otherwise
V: Cleared
C: Cleared

Description: The source character string is searched from most significant to least significant character until the first occurrence of a character which is not a member of the character set. A character-string descriptor is returned in R0,R1 which represents the portion of the source character string beginning with the character which is not a member of the character set. If the source character string contains only characters which are in the character set, the instruction returns a vacant character-string descriptor with an address one greater than that of the least significant character of the source character string. The condition codes reflect the resulting value in R0.

When the instruction is completed, R0,R1 contain a character-string descriptor which represents the substring of the source character string beginning with the most significant character which is not a member of the character set.

Notes:

1. If the initial source character string descriptor is vacant, the instruction terminates with the condition codes indicating that only characters in the set were found. The original source character string descriptor is returned in R0,R1.
2. The source character string and character set table may overlap in any way.
3. The condition codes will be set as if this instruction were followed by TST R0.
4. The effect of the instruction is *unpredictable* if the entire 256-byte character set table is not in readable memory.

SUBN	**SUBtract Numeric decimal**	**076051_8**
SUBP	**SUBtract Packed decimal**	**076071_8**

Operands: R0,R1: Source 1 Descriptor
R2,R3: Source 2 Descriptor
R4,R5: Destination Descriptor

SUBNI	**SUBtract Numeric decimal (In-line)**	**076151_8**
SUBPI	**SUBtract Packed decimal (In-line)**	**076171_8**

Operands: addr + 2: pointer to Source 1 Descriptor
addr + 4: pointer to Source 2 Descriptor
addr + 6: pointer to Destination Descriptor

Operation: (dst) ← (src2) - (src1)
(R0),(R1),(R2),(R3) ← 0 (SUBN, SUBP only)

Condition Codes:
N: Set if dst < 0; cleared otherwise
Z: Set if dst = 0; cleared otherwise
V: Set if dst cannot contain all significant digits of the result; cleared otherwise
C: Cleared

Description: Src1 is subtracted from src2, and the result is stored in the destination string. The condition codes reflect the value stored in the destination string and whether all significant digits were stored.

When the instruction is completed, the source descriptor registers (R0–R3) are cleared (SUBP, SUPM only).

Notes:

1. The operation of these instructions is unaffected by any overlap of the source strings provided that each source string is a valid representation of the specified data type.
2. Source strings may overlap the destination string only if all corresponding digits of the strings are in coincident bytes in memory.

APPENDIX G
PDP-11 FAMILY DIFFERENCES TABLE

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

Since the instruction set for all F-11 based processors is identical, the 23/24 column refers to the PDP-11/23 PLUS, the PDP-11/24, the LSI-11/23, the MICRO/PDP-11, and the F-11 chip itself.

The LSI-11 column includes the LSI-11/2.

The T-11 column also refers to the FALCON SBC-11/21.

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

PROCESSORS

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
1. OPR %R, (R) +; OPR %R, – (R) using the same register as both source and destination: contents of R are incremented (decremented) by 2 before being used as the source operand	X						X	X			X	X	X	
OPR %R, (R) +; OPR %R, – (R) using the same register as both register and destination: initial contents of R are used as the source operand.		X	X	X	X	X			X	X				X
2. OPR %R, @ (R) +; OPR %R, @ – (R) using the same register as both source and destination: contents of R are incremented (decremented) by 2 before being used as the source operand.	X						X	X			X	X	X	
OPR %R, @ (R) +; OPR %R, @ – (R) using the same register as both source and destination: initial contents of R are used as the source operand.		X	X	X	X	X			X	X				X
3. OPR PC, X (R); OPR PC, @ X (R); OPR PC, @ A; OPR PC, A: location A will contain the PC of OPR + 4.	X						X	X			X	X	X	
OPR PC, X (R); OPR PC, @ X (R), OPR PC, A; OPR PC, @ A: location A will contain the PC of OPR + 2.		X	X	X	X	X			X	X				X
4. JMP (R) + or JSR reg, (R) +: contents of R are incremented by 2, then used as the new PC address.						X	X							
JMP (R) + or JSR reg, (R) +: initial contents of R are used as the new PC.	X	X	X	X	X			X	X	X	X	X	X	X

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
5. JMP %R or JSR reg, %R traps to 10 (illegal instruction).	X		X	X	X	X	X	X			X		X	NA
JMP %R or JSR reg, %R traps to 4 (illegal instruction).		X							X	X		X		NA
6. SWAB does *not* change V.							X							
SWAB clears V.	X	X	X	X	X	X		X	X	X	X	X	X	X
7. Register addresses (177700–177717) are valid program addresses when used by CPU.						X							—[1]	—[1]
Register addresses (177700–177717) time out when used as a program address by the CPU. Can be addressed under console operation.		X	X	X			X	X	X	X	X			NA
Register addresses (177700–177717) time out when used as an address by CPU or console.	X				X							X		
8. Basic instructions noted in PDP-11 processor handbook.	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SOB, MARK, RTT, SXT instructions*	X	X		X	X			X	X	X	X	X	X	—[2]
ASH, ASHC, DIV, MUL, XOR	X	X		X	X			X	X	X	X	X		X
Floating Point instructions in base machine.											X	X		
MFPT Instruction.	X	X										X		
The external option KE11-A provides MUL, DIV, SHIFT operation in the same data format.						X	X							

* RTT instruction is available in 11/04 but is different than other implementations

[1] Register addresses (177700–177717) are handled as regular memory addresses in the I/O page.
[2] All but MARK.

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
The KE11-E (Expansion Instruction Set) provides the instructions MUL, DIV, ASH, and ASHC. These new instructions are 11/45 compatible.								X						
The KE11-F (Floating Instruction Set) adds unique stack ordered oriented point instructions: FADD, FSUB, FMUL, FDIV.								X						
The KEV-11 adds EIS/FIS instructions					X									
MFP, MTP instructions	X	X		X				X		X	X	X		
SPL Instruction		X							X	X		X		
CSM Instruction		X										X		
9. Power fail during RESET instruction is not recognized until after the instruction is finished (70 milliseconds). RESET instruction consists of 70 millisecond pause with INIT occurring during first 20 milliseconds.							X	X			X			
Power fail immediately ends the RESET instruction and traps if an INIT is in progress. A minimum INIT of 1 microsecond occurs if instruction aborted. PDP11-04/34/44 are similar with no minimum INIT time.		X	X	X					X	X				
Power fail acts the same as 11/45 (22 milliseconds with about 300 nanoseconds minimum). Power fail during RESET fetch is fatal with no power down sequence.						X								

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
RESET instruction consists of 10 microseconds of INIT followed by a 90 microsecond pause Reset instruction consists of a minimum 8 4 microseconds followed by a minimum 100 nanosecond pause. Power fail not recognized until the instruction completes.	X				X							X		
10 No RTT instruction If RTT sets the "T" bit, the "T" bit trap occurs after the instruction following RTT.	X	X	X	X	X	X	X	X	X	X	X	X	X	X
11 If RTI sets "T" bit, "T" bit trap is acknowledged after instruction following RTI If RTI sets "T" bit, "T" bit trap is acknowledged immediately following RTI.	X	X	X	X	X	X	X	X	X	X	X	X	X	X
12 If an interrupt occurs during an instruction that has the "T" bit set, the "T" bit trap is acknowledged before the interrupt If an interrupt occurs during an instruction and the "T" bit is set, the interrupt is acknowledged before "T" bit trap.	X	X	X	X	X	X	X	X	X	X	X	X	X	NA[1] NA
13. "T" bit trap will sequence out of WAIT instruction "T" bit trap will not sequence out of WAIT instruction. Waits until an interrupt	X	X	X	X	X	X	X	X	X	X	X	X	X	NA

[1] Interrupts not visible to VAX compatibility mode

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
14. Explicit reference (direct access) to PS can load "T" bit. Console can also load "T" bit.			X			X	X							
Only implicit references (RTI, RTT, traps and interrupts) can load "T" bit. Console cannot load "T" bit.	X	X		X	X			X	X	X	X	X	X	X
15. Odd address/non-existent references using the SP cause a HALT. This is a case of double bus error with the second error occurring in the trap servicing the first error. Odd address trap not implemented in LSI-11, 11/23 or 11/24.		X	X	X	X	X	X							
Odd address/non-existent references using the stack pointer cause a fatal trap. On bus error in trap service, new stack created at 0/2.	X							X	X	X	X	X	—[1]	—[2]
16. The first instruction in an interrupt routine will not be executed if another interrupt occurs at a higher priority level than assumed by the first interrupt.	X	X	X	X	X	X		X	X	X	X	X	X	X
The first interrupt in an interrupt service is guaranteed to be executed.							X							
17. Single general purpose register set implemented.	X	X	X	X	X	X	X	X			X		X	X
Dual general purpose register set implemented.									X	X		X		

[1] Odd address/non-existent references using SP do not trap.
[2] Odd address aborts to native mode

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
18. PSW address, 177776, not implemented; must use instructions MTPS (move to PS) and MFPS (move from PS).					X								X	—[3]
PSW address implemented, MTPS and MFPS not implemented.		X	X			X	X	X	X	X	X			
PSW address and MTPS and MFPS implemented.	X			X								X		
19. Only one interrupt level (BR4) exists.					X									
Four interrupt levels exist.	X	X	X	X		X	X	X	X	X	X	X	X	NA
20. Stack overflow not implemented.					X								X	X
Some sort of stack overflow implemented.	X	X	X	X		X	X	X	X	X	X	X		
21. Odd address trap not implemented.	X				X								X	
Odd address trap implemented.		X	X	X		X	X	X	X	X	X	X		X
22. FMUL and FDIV instructions implicity use R6 (one push and pop); hence R6 must be set up correctly.					X									
FMUL and FDIV instructions do not implicitly use R6.								X						NA
23. Due to their execution time, EIS instructions can abort because of a device interrupt.					X									X
EIS instructions do not abort because of a device interrupt.	X	X		X				X	X	X	X	X		NA
24. Due to their execution time, FIS instructions can abort because of a device interrupt.					X			X						NA

[3] Can reference PSW only from native mode.

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
34. Opcodes 75040 thru 75777 trap to 10 as reserved instructions.	X	X	X	X		X	X	X	X	X	X	X	X	—[1]
If KEV-11 options is present, opcodes 75040 thru 75577 can be used as escapes to user microcode. If no user microcode exists, a trap to 10 occurs.					X									
35. Opcodes 170000 thru 177777 trap to 10 as reserved instructions.			X			X	X	X					X	—[1]
Opcodes 170000 thru 177777 are implemented as floating point instructions.	X	X		X					X	X	X	X		
Opcodes 170000 thru 177777 can be used as escapes to user microcode. If no user microcode exists, a trap to 10 occurs.					X									
Opcode 076600 used for maintenance.											X			
36. CLR and SXT do just a DATO sequence for the last bus cycle.	X											X		—[1]
CLR and SXT do DATIP-DATO sequence for the last bus cycle.		X	X	X	X	X	X	X	X	X	X		—[2]	
37. MEM MGT maintenance mode MMR0 bit 8 is implemented.		X		X				X	X	X	X			
MEM MGT maintenance mode MMR0 bit 8 is not implemented.	X											X		NA
38. PS<15:12>, non-kernel mode, non-kernel stack pointer and MTPx and MFPx instructions exist even when MEM MGT is not configured.	X	X							X	X	X	X		

[1] Traps to native mode.

[1] Unpredictable.
[2] CLR and SXT do DATI-DATO.

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
PS<15:12>, non-kernel mode, non-kernel stack pointer, and MTPx and MFPx instructions exist only when MEM MGT is configured.								X						NA
39. Current mode PS bits <15:14> set to 01 or 10 will cause a MEM MGT trap upon any memory reference.	X			X				X				X		
Current mode PS bits <15:14> set to 10 will be treated as kernel mode (00) and not cause a MEM MGT trap.	X													NA
Current mode PS bits <15:14> set to 10 will cause a MEM MGT trap upon any memory reference.		X							X	X				
40. MTPS in user mode will cause MEM MGT trap if PS address 177776 not mapped. If mapped, PS <7:5> and <3:0> affected.				X										
MTPS in non-user mode will not cause MEM MGT trap and will only affect PS <3:0> regardless of whether PS address 177776 is mapped.	X											X		NA
41. MFPS in user mode will cause MEM MGT if PS address 177776 not mapped. If mapped, PS <7:0> are accessed.				X										
MTPS in user mode will not trap regardless of whether PS address 177776 is mapped.	X											X		NA

[1] Unpredictable
[2] CLR and SXT do DATI-DATO

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
25. Due to their execution time, FP11 instructions can abort because of a device interrupt.*	X													
FP11 instructions do not abort because of a device interrupt.		X		X					X	X	X	X		NA
26. EIS instructions do a DATIP and DATO bus sequence when fetching source operand.					X									
EIS instructions do a DATI bus sequence when fetching source operand.	X	X		X				X	X	X	X	X		NA
27. MOV instruction does just a DATO bus sequence for the last memory cycle.	X	X		X	X			X	X	X	X	X		—[1]
MOV instruction does a DATIP and DATO bus sequence for the last memory cycle.			X			X	X						—[2]	
28. If PC contains non-existent memory and a bus error occurs, PC will have been incremented.	X	X	X	X	X	X	X		X	X		X		
If PC contains non-existent memory address and a bus error occurs, PC will be unchanged.								X					—[3]	X
29. If register contains non-existent memory address in mode 2 and a bus error occurs, register will be incremented.	X				X	X	X	X	X	X		X	—[3]	
Same as above but register is unchanged.		X	X	X										

* Integral floating point assumed on 11/23 and 11/24; FP11E assumed for 11/60.
[1] Implementation dependent.
[2] MOV instruction does a DATI and a DATO bus sequence for last memory cycle.
[3] Does not support bus errors.

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
30. If register contains an odd value in mode 2 and a bus error occurs, register will be incremented.	X				X			X	X	X		X	—[3]	—[4]
If register contains an odd value in mode 2 and a bus error occurs, register will be unchanged.		X	X	X		X	X							
31. Condition codes restored to original values after FIS interrupt abort (EIS doesn't abort on 35/40).								X						
Condition codes that are restored after EIS/FIS interrupt abort are indeterminate.					X									NA
32. Opcodes 075040 through 075377 unconditionally trap to 10 as reserved opcodes.	X	X	X	X		X	X	X	X	X	X	X	X	—[1]
If KEV-11 option is present, opcodes 75040 through 07533 perform a memory read using the register specified by the low order 3 bits as a pointer. If the register contents are a non-existent address, a trap to 4 occurs. If the register contents are an existent address, a trap to 10 occurs.					X									
33. Opcodes 210 thru 217 trap to 10 as reserved instructions.	X	X	X	X		X	X	X	X	X	X	X	X	—[1]
Opcodes 210 thru 217 are used as a maintenance instruction.					X									

[3] Does not support bus errors.
[4] Unpredictable
[1] Traps to native mode

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
42. Programs cannot execute out of internal processor registers.												X		
Programs can execute out of internal processor registers.	X	X		X				X	X	X	X			
43. A HALT instruction in user or supervisor mode will trap thru location 4.		X							X	X		X		
A HALT instruction in user or supervisor mode will trap thru location 10	X			X				X			X		—[1]	—[2]
44. PDR bit <0> implemented.									X	X				
PDR bit <0> not implemented.	X	X		X				X			X	X	X	X
45. PDR bit <7> (any access) implemented.									X	X				
PDR bit <7> (any access) not implemented.	X	X		X				X			X	X	X	X
46. Full PAR <15:0> implemented.	X	X								X		X		
Only PAR <11:0> implemented.				X				X	X		X		X	X
47. MMR0<12>—trap-memory management—implemented.									X	X				
MMR0<12> not implemented.	X	X		X				X			X	X	X	X
48. MMR3<2:0>—D space enable—implemented.		X							X	X		X		
MMR3<2:0> not implemented.	X			X				X			X		X	X
49. MMR3<5:4>—IOMAP, 22-bit mapping enabled—implemented.	X	X								X		X		
MMR3<5:4> not implemented.				X				X	X		X		X	X

[1] HALT pushes PC & PSW to stack, loads PS with 340 and PC with <powerup address> + 40
[2] Traps to native mode.

ITEM	23/24	44	04	34	LSI11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
50. MMR3<3>—CSM enable—implemented.		X										X		
MMR3<3> not implemented.	X			X				X	X	X	X		X	X
51. MMR2 tracks instruction fetches and interrupt vectors.									X	X				
MMR2 tracks only instruction fetches.	X	X		X				X			X	X	NA	NA
52. MFPx %6, MTPx when PS<13:12>= 10 gives unpredictable results.	X	X		X				X	X	X	X			
MTPx %6, MTPx %6 when PS<13:12>= 10 uses user stack pointer.												X	NA	NA

[1] HALT pushes PC & PSW to stack, loads PS with 340 and PC with <powerup address> + 40.
[2] Traps to native mode.

APPENDIX H
SOFTWARE DISTRIBUTION MEDIA

There are three basic requirements for running a given software product on a system:

1. Compatibility—The software code must work with the CPU, storage devices, and interfaces.
2. Media—The software code must be on a removable storage medium that the target machine can accept.
3. License—The cost of DIGITAL software generally represents the right to run that code on a single system and, in addition, may include support.

COMPATIBILITY

The following operating systems support MICRO/PDP-11:

- RT-11 Version 5
- RSX-11M Version 4.2
- RSX-11M-PLUS Version 2.2
- RSTS/E Version 8
- CTS-300 Version 8
- DSM-11 Version 7
- V7M-11 (UNIX/TM)

In general, software products that run with these operating systems are also compatible with MICRO/PDP-11.

MEDIA

There are currently two removable media used for software distribution to the MICRO/PDP-11: the RX50 diskette and the RL02 cartridge disk. Software distribution of operating systems is:

RT-11	RX50, RL02
RSX-11M	RL02
RSX-11M-PLUS	RL02

RSTS/E	RL02
CTS-300	RX50, RL02
DSM-11	RL02
V7M-11	RX50, RL02
MicroPower/Pascal	RX50, RL02

In general, languages, network software, and other software products are not distributed on RX50 diskettes.

To develop an application using software not distributed on RX50 diskettes, your development system will need an RL02 disk drive (RLV22-AP with controller for a MICRO/PDP-11). The development system can build the application on RX50 diskettes for transfer to a target MICRO/PDP-11 without an RL02.

LICENSE

A license is required for each processor using a DIGITAL software product.

APPENDIX I
ODT

ODT commands allow you to access memory and registers, run diagnostics, and monitor the system.

Table I-1 Console ODT Commands*

Command	Symbol	Description
Slash	/	Prints the contents of a specified location
Carriage return	<CR>	Closes an open location
Line feed	<LF>	Closes an open location and then opens the next contiguous location
Internal register designator	$ or R	Opens a specific processor register
Processor status word designator	S	Opens the PS, must follow a "$" or "R" command
Go	G	Starts program execution
Proceed	P	Resumes execution of a program
Binary dump	Control-Shift-S	Manufacturing use only

* On the F-11-based MICRO/PDP-11, all addresses in ODT must be 18-bit addresses between 0 and 777776_8.

APPENDIX J
SEVEN-BIT ASCII CODE

Octal Code	Char	Octal Code	Char	Octal Code	Char	Octal Code	Char
000	NUL	040	SP	100	@	140	`
001	SOH	041	!	101	A	141	a
002	STX	042	"	102	B	142	b
003	ETX	043	#	103	C	143	c
004	EOT	044	$	104	D	144	d
005	ENQ	054	%	105	E	145	e
006	ACK	046	&	106	F	146	f
007	BEL	047	'	107	G	147	g
010	BS	050	(	110	H	150	h
011	HT	051	)	111	I	151	i
012	LF	052	*	112	J	152	j
013	VT	053	+	113	K	153	k
014	FF	054	,	114	L	154	l
015	CR	055	-	115	M	155	m
016	SO	056	.	116	N	156	n
017	SI	057	/	117	O	157	o
020	DLE	060	0	120	P	160	p
021	DC1	061	1	121	Q	161	q
022	DC2	062	2	122	R	162	r
023	DC3	063	3	123	S	163	s
024	DC4	064	4	124	T	164	t
025	NAK	065	5	125	U	165	u
026	SYN	066	6	126	V	166	v
027	ETB	067	7	127	W	167	w
030	CAN	070	8	130	X	170	x
031	EM	071	9	131	Y	171	y
032	SUB	072	:	132	Z	172	z
033	ESC	073	;	133	[	173	{
034	FS	074	<	134	\	174	|
035	GS	075	=	135	] or ↑	175	}
036	RS	076	>	136	^	176	~
037	US	077	?	137	— or ←	177	DEL

APPENDIX K
FOR MORE INFORMATION . . .

This handbook is intended to provide the reader with an overview of the MICRO/PDP-11 family and related products. Many readers, however, require information that is either more detailed or wider in scope. For this reason, DIGITAL publishes a complete spectrum of documentation, with subjects ranging from networking concepts to detailed circuit descriptions.

The following tables contain the titles and associated DIGITAL part numbers of documents that apply to the basic MICRO/PDP-11. Appendix C contains the titles and part numbers of documents that apply to the MICRO/PDP-11 options.

Unless otherwise specified, the documents in these lists can be ordered from DIGITAL's Accessories and Supplies Group, P.O. Box CS2008, Nashua, NH 03061. Rush orders can be placed by telephone to DIGITAL's toll-free hotline: (800) 258-1710.

Table K-1 Documents of General Interest

Document Name	Part Number
Introduction to Local Area Networks	EB-22714-18
Computer Engineering: A DEC View of Hardware Systems Design	EY-BX007-DP*
Technical Aspects of Data Communication	EY-BX006-DP*

* These items are available from:
Digital Press
12A Esquire Road
Billerica, MA 01862
Telephone: (800) 343-8321

Table K-2 Hardware and Software Handbooks

Document Name	Part Number
PDP-11 Architecture Handbook	EB-23657-18
PDP-11 Microcomputer Interfaces Handbook	EB-23144-18
PDP-11 Microcomputers and Memories Handbook	EB-18451-20
PDP-11 Software Handbook	EB-08687-20
PDP-11 Processor Handbook	EB-19402-20
Terminals and Communications Handbook	EB-20750-20
Peripherals Handbook	EB-20443-20
Maintenance Aids Handbook	EB-20174-75
Spares Kit Handbook	EB-24015-75
Cables Handbook	EB-19187-75
Documentation Kit Handbook	EB-19769-75

Table K-3 MICRO/PDP-11 Hardware Manuals

Document Name	Part Number
MICRO/PDP-11 System Owner's Manual	EK-OLCP5-OM
MICRO/PDP-11 System Technical Manual	EK-OLCP5-TM
MICRO/PDP-11 Site Preparation and Verification Guide	EK-OLCP5-SP
MICRO/PDP-11 Unpacking and Installation Guide	EK-OLCP5-IN
MICRO/PDP-11 Pocket Service Guide	EK-OLCP5-PS
MICRO/PDP-11 Illustrated Parts Breakdown	EK-OLCP5-IP
MICRO/PDP-11 Option Manual	EK-OLCP5-OD
RQDX1 Controller Module User's Guide	EK-RQDX1-UG
KDF11-BA CPU Module User's Guide	EK-KDFEB-UG
KDF11-B Field Maintenance Print Set	MP-01236
MSV11-P User's Guide	EK-MSVOP-UG
MSV11-P Field Maintenance Print Set	MP-02139

Catalogs and System Software Descriptions

DECdirect Catalog

DECdirect System Builder Catalog

Documentation Products Directory

PDP-11 Systems and Options Catalog

DSM-11 Technical Summary

RSTS/E Technical Summary

RSX-11M Technical Summary

RT-11 Technical Summary

PDP-11 Software Source Book

These items are available from your local DIGITAL sales representative.

INDEX

MICRO/PDP-11 HANDBOOK 1983–84

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. _____

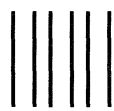
Address _____

City _____ State _____ Zip _____

(staple here)

(staple here)

(please fold here)



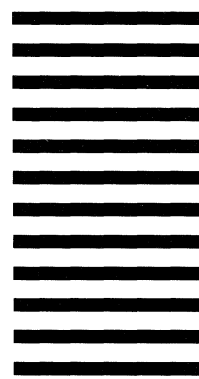
No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 33 MAYNARD, MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
NEW PRODUCTS MARKETING
PK3-1/M92
MAYNARD, MASS. 01754



HANDBOOK SERIES

Microcomputers and Memories
Microcomputer Interfaces
MICRO/PDP-11
PDP-11 Processor
PDP-11 Architecture
PDP-11 Software
Peripherals
Terminals and Communications
VAX Architecture
VAX Software
VAX Hardware

DIGITAL EQUIPMENT CORPORATION, Corporate Headquarters: Maynard, MA 01754, Tel. (617) 897-5111 — SALES AND SERVICE OFFICES; UNITED STATES — ALABAMA, Birmingham, Huntsville ARIZONA, Phoenix, Tucson ARKANSAS, Little Rock CALIFORNIA, Costa Mesa, El Segundo, Los Angeles, Modesto, Monrovia, Oakland, Pasadena, Sacramento, San Diego, San Francisco, Santa Barbara, Santa Clara, Santa Monica, Sherman Oaks, Sunnyvale, Torrance COLORADO, Colorado Springs, Denver CONNECTICUT, Fairfield, Meriden DELAWARE, Newark, Wilmington FLORIDA, Jacksonville, Melbourne, Miami, Orlando, Pensacola, Tampa GEORGIA, Atlanta HAWAII, Honolulu IDAHO, Boise ILLINOIS, Chicago, Peoria INDIANA, Indianapolis IOWA, Bettendorf KENTUCKY, Louisville LOUISIANA, Baton Rouge, New Orleans MAINE, Portland MARYLAND, Baltimore, Odenton MASSACHUSETTS, Boston, Burlington, Springfield, Waltham MICHIGAN, Detroit, Kalamazoo MINNESOTA, Minneapolis MISSOURI, Kansas City, St. Louis NEBRASKA, Omaha NEVADA, Las Vegas, Reno NEW HAMPSHIRE, Manchester NEW JERSEY, Cherry Hill, Parsippany, Princeton, Somerset NEW MEXICO, Albuquerque, Los Alamos NEW YORK, Albany, Buffalo, Long Island, New York City, Rochester, Syracuse, Westchester NORTH CAROLINA, Chapel Hill, Charlotte OHIO, Cincinnati, Cleveland, Columbus, Dayton OKLAHOMA, Tulsa OREGON, Eugene, Portland PENNSYLVANIA, Allentown, Harrisburg, Philadelphia, Pittsburgh RHODE ISLAND, Providence SOUTH CAROLINA, Columbia, Greenville TENNESSEE, Knoxville, Memphis, Nashville TEXAS, Austin, Dallas, El Paso, Houston, San Antonio UTAH, Salt Lake City VERMONT, Burlington VIRGINIA, Arlington, Lynchburg, Norfolk, Richmond WASHINGTON, Seattle, Spokane WASHINGTON D.C. WEST VIRGINIA, Charleston WISCONSIN, Madison, Milwaukee INTERNATIONAL — EUROPEAN AREA HEADQUARTERS: Geneva, Tel: [41] (22)-93-33-11 INTERNATIONAL AREA HEADQUARTERS: Acton, MA 01754, U.S.A., Tel: (617) 263-6000 ARGENTINA, Buenos Aires AUSTRALIA, Adelaide, Brisbane, Canberra, Darwin, Hobart, Melbourne, Newcastle, Perth, Sydney, Townsville, Victoria AUSTRIA, Vienna BELGIUM, Brussels BRAZIL, Rio de Janeiro, Sao Paulo CANADA, Calgary, Edmonton, Hamilton, Halifax, Kingston, London, Montreal, Ottawa, Quebec City, Regina, Toronto, Vancouver, Victoria, Winnipeg CHILE, Santiago COLOMBIA, Bogota DENMARK, Copenhagen EGYPT, Cairo ENGLAND, Basingstoke, Birmingham, Bristol, Ealing, Epsom, Leeds, Leicester, London, Manchester, Newmarket, Reading, Welwyn FINLAND, Helsinki FRANCE, Bordeaux, Lille, Lyon, Marseille, Nantes, Paris, Puteaux, Strasbourg HONG KONG INDIA, Bangalore, Bombay, Calcutta, Hyderabad, New Delhi IRELAND, Dublin ISRAEL, Tel Aviv ITALY, Milan, Padova, Rome, Turin JAPAN, Fukuoka, Nagoya, Osaka, Tokyo, Yokohama KOREA, Seoul KUWAIT, Safat MEXICO, Mexico City, Monterrey NETHERLANDS, Amsterdam, The Hague, Utrecht NEW ZEALAND, Auckland, Christchurch, Wellington NIGERIA, Lagos NORTHERN IRELAND, Belfast NORWAY, Oslo, PERU, Lima PUERTO RICO, San Juan SAUDI ARABIA, Jeddah SCOTLAND, Edinburgh REPUBLIC OF SINGAPORE, SPAIN, Barcelona, Madrid SWEDEN, Gothenburg, Malmoe, Stockholm SWITZERLAND, Geneva, Zurich TAIWAN, Taipei TRINIDAD, Port of Spain VENEZUELA, Caracas WEST GERMANY, Berlin, Cologne, Frankfurt, Hamburg, Hannover, Munich, Nuremberg, Stuttgart YUGOSLAVIA, Belgrade, Ljubljana, Zagreb

ORDER CODE: EB-24944-18

PRINTED IN U.S.A. EB 24944 18/83 COPYRIGHT © 1983 DIGITAL EQUIPMENT CORPORATION ALL RIGHTS RESERVED