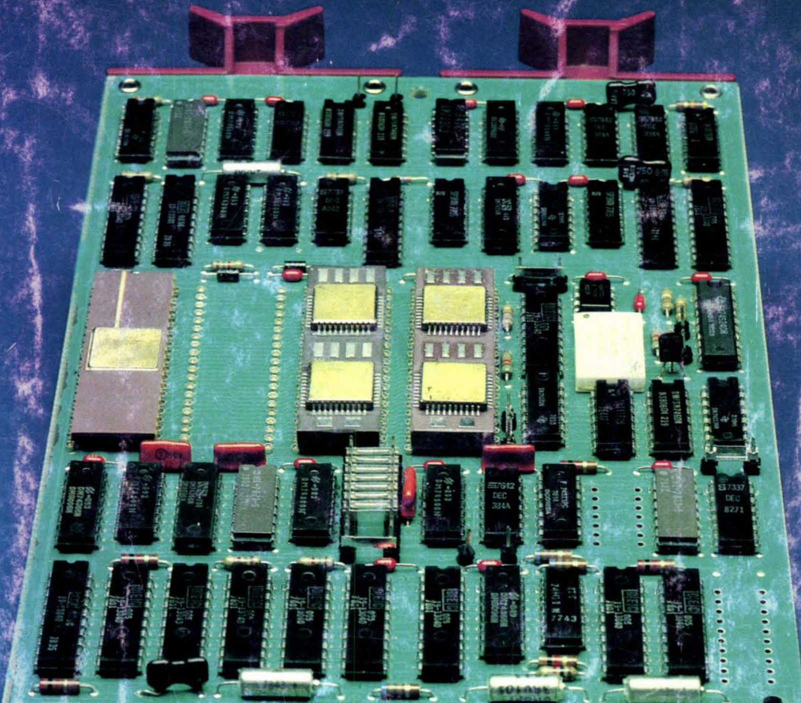
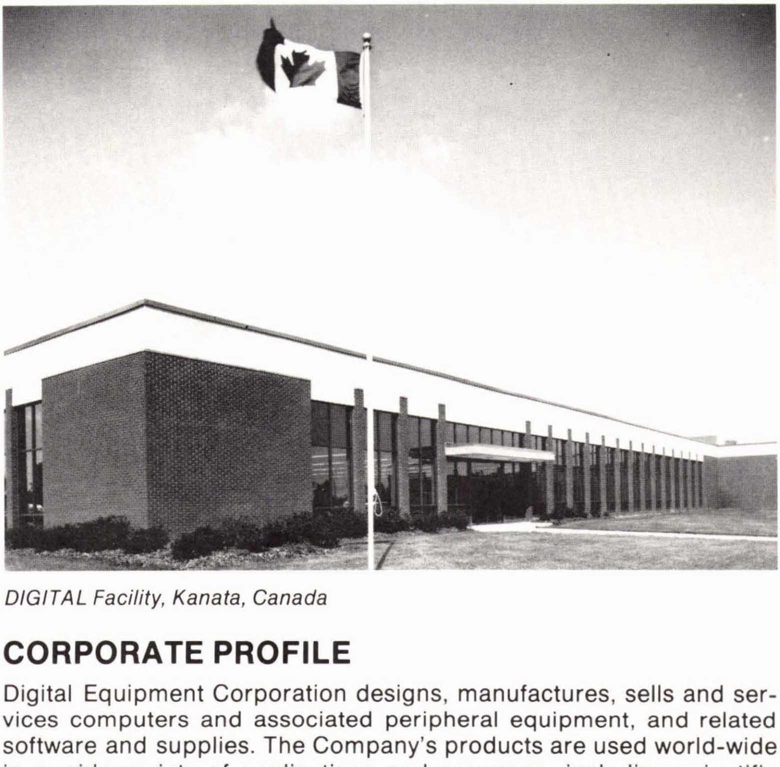
INCLUDING MEMORIES

digital

microcomputer processor handbook




DIGITAL Facility, Kanata, Canada

CORPORATE PROFILE

Digital Equipment Corporation designs, manufactures, sells and services computers and associated peripheral equipment, and related software and supplies. The Company's products are used world-wide in a wide variety of applications and programs, including scientific research, computation, communications, education, data analysis, industrial control, timesharing, commercial data processing, word processing, health care, instrumentation, engineering and simulation.

microcomputer processor handbook

digital

CONTENTS

digital PDP 11/23

CHAPTER 1
INTRODUCTION

DIGITAL's development of microcomputers began in 1975, with the introduction of the LSI-11. Large Scale Integration technological advances in semiconductor chip design have led to rapid progress in the reduction of cost and escalation of functionality.

This handbook discusses three members of DIGITAL's LSI-11 processor family—the LSI-11 (KD11-F), LSI-11/2 (KD11-HA) and the newest member, the LSI-11/23 (KDF11-AA).

The original **LSI-11** is packaged on a *quad-height* board, 10″ × 8.9″ (25.4 cm × 22.8 cm). In 1977, DIGITAL introduced the **LSI-11/2**, a smaller, *double height* version of the LSI-11. The LSI-11/2 employs the same chip set as the original LSI-11, but lacks the 8K bytes of memory contained on the quad height board. The smaller physical size of the LSI-11/2 made it convenient for incorporation into instrumentation devices.

With the introduction of the LSI-11/2, DIGITAL announced new memory and communications peripheral boards in the double height form factor.

The newest member of the LSI-11 family, the **LSI-11/23** was announced in 1979. Software- and hardware-compatible with LSI-11 processors, it's improved functionality provides minicomputer power with the price and size benefits of a microcomputer.

The most fundamental LSI-11/23 design decision was to remain with the LSI-11 bus architecture. This architecture assures the continued hardware and software commonality of LSI-11 family. Expansion potential was incorporated into the original design of the LSI-11 bus. The original design specified 18 memory address lines, the upper two reserved for later use. These high order address lines are now utilized by the LSI-11/23 to permit increased memory addressing capability.

The size and hardware compatibility of the LSI-11/23 and the other LSI-11 family members permits substitution of the newer module into existing applications requiring greater functionality and performance. It continues this improvement by permitting development of products that can be used with all LSI-11 processors. One such device is the MXV11 multifunction board, a double height board containing either 8K or 32K bytes of RAM, sufficient ROM/PROM for bootstrap or program functions, a crystal clock, and two serial input/output ports. When used together, the LSI-11/23 processor board and the multifunction board contain all the functions needed to implement a complete small system.

When an LSI processor board is packaged with a backplane, power supply and rack mountable box, DIGITAL considers it a member of the outstanding PDP-11 family. An LSI-11 or LSI-11/2 boxed with backplane, memory, and power supply, becomes PDP-11/03 or PDP-11/03-Ls and the LSI-11/23 boxed with backplane, memory, and power supply, becomes a PDP-11/23. These *low end* members of the PDP-11 family have, with a very few exceptions, all the features and benefits of the other family members.

- **Extensive Computer Power and Small Processor Size**
  The processor modules are built around a set of N-channel metal oxide semiconductor (MOS) chips, which include control and data elements as well as microcoded read-only memory (microms). The memory is microprogrammed to emulate the powerful PDP-11 instruction set, and also contains routines for on-line debugging techniques (ODT) that function as a console emulator. The processor also contains a 16-bit buffered parallel input/output (I/O) bus, a real-time clock input, priority interrupt control logic, bus arbitration logic, power-fail/auto restart, and other features to provide stand-alone operation. LSI-11 processors are contained on three basic module types to suit a variety of applications.
- **Modularity**
  The processor, memory, device interfaces, backplane, and interconnecting hardware are all modular in design. Module selection, such as the type and size of memory and device interfaces, enables custom tailoring to meet specific applications requirements.
- **16-Bit Word** (Two 8-Bit Bytes)
- **Word or Byte Processing**
  Efficient handling of 8-bit characters without the need to rotate, swap, or mask.
- **Asynchronous Operation**
  LSI-11 processor and system components (memory and peripherals) run at their highest possible speed.
- **Stack Processing**
  Hardware sequential memory manipulation makes it easy to handle structured data, subroutines, and interrupts.
- **Direct Memory Access (DMA)**
  Direct memory access for multiple devices is inherent in the architecture.
- **Eight General-Purpose Registers**
  For accumulators or address generation.
- **Priority-Structured I/O System**
  Daisy-chained grant signals provide a priority-structured I/O system.

- **Vectored Interrupts**
  Fast interrupt response without device polling.
- **Single and Double Operand Instructions**
  Powerful and efficient set of programming instructions.
- **Power-Fail/Auto Restart**
  Whenever dc power sequencing signals indicate an impending ac power loss, a microcoded power-fail sequence is initiated. When power is restored, the processor can automatically return to the run state (when nonvolatile memory is contained in the system).
- **A Wide Variety of Options**
  Options include memory (read/write and read-only); serial line, parallel line, and DMA interfaces; mass storage devices (hard disk and floppy disk systems); analog interfaces (D/A and A/D converters); bus expansion, power supply, and hardware options (backplanes, boxes, etc.).

SYSTEM ARCHITECTURE

A complete and powerful microcomputer system can be configured using an LSI-11 microcomputer, appropriate memory, I/O devices, and interconnection hardware. The LSI-11 Bus provides communication between system components. A typical system configuration is shown in the accompanying figure.

All modules connected to this common LSI-11 Bus structure receive the same interface signals. LSI-11 Bus control and data lines are open-collector lines which are asserted when low. All data and most control lines are bidirectional. All transactions on the bus are asynchronous. The LSI-11 processors use the following LSI-11 Bus signals: 16 multiplexed data/address lines, 2 multiplexed address/parity lines, 6 data transfer control lines, 6 system control lines, and 8 interrupt and direct memory access (DMA) control lines.

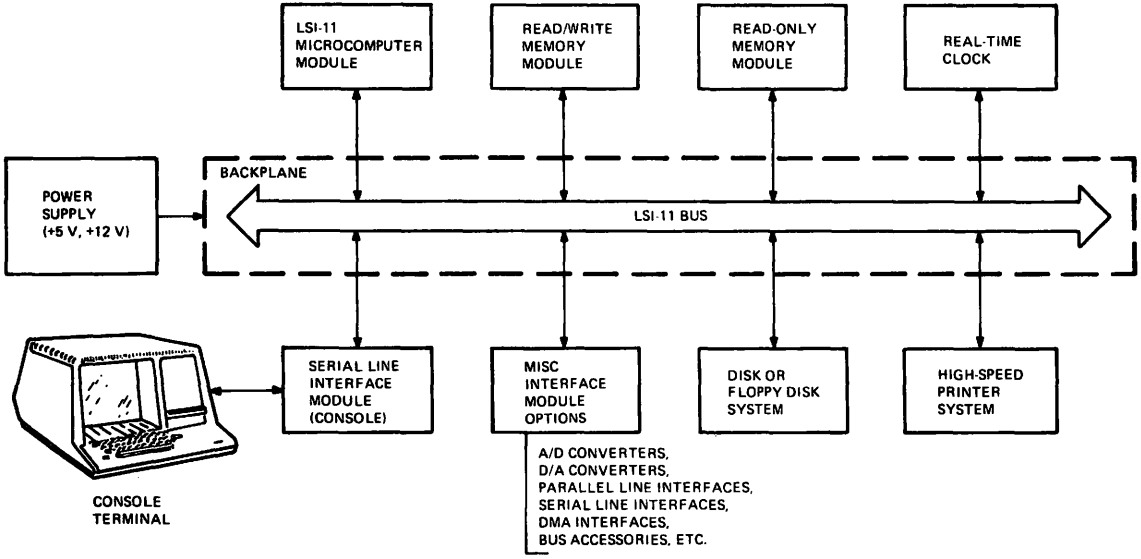
Interrupt and DMA are implemented with two daisy-chained grant signals which provide a priority-structured I/O system. The highest priority device is the module located electrically closest to the microcomputer module. A device passes grant signals to lower priority devices only when it is not requesting service.

The LSI-11 bus provides a vectored interrupt interface for any device. Device polling is not required in processing interrupt requests. When an interrupting device receives a grant, the device passes an interrupt vector to the processor, which points to a new processor status word and the starting address of an interrupt service routine for the device.

LSI-11 backplane options contain all LSI-11 bus wiring plus power distribution wiring to all device locations.

THE LSI-11 MICROCOMPUTER

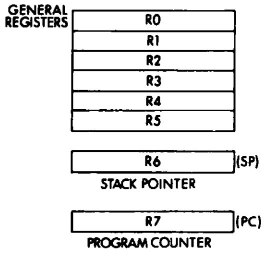
The microcomputer connected to the LSI-11 bus controls the time allocation of the LSI-11 Bus for peripherals and performs arithmetic and logic operations and instruction decoding. It contains multiple high-speed, general-purpose registers which can be used as accumulators, address pointers, index registers, and for other specialized functions. The processor does both single and double operand instruction and handles both 16-bit word and 8-bit byte data. The bus permits DMA data transfers directly between I/O devices and memory without disturbing the processor registers.


LSI-11 MICROCOMPUTER MODULE
READ/WRITE MEMORY MODULE
READ-ONLY MEMORY MODULE
REAL-TIME CLOCK
POWER SUPPLY (+5 V, +12 V)
BACKPLANE
LSI-11 BUS
SERIAL LINE INTERFACE MODULE (CONSOLE)
MISC INTERFACE MODULE OPTIONS
DISK OR FLOPPY DISK SYSTEM
HIGH-SPEED PRINTER SYSTEM
CONSOLE TERMINAL
A/D CONVERTERS,
D/A CONVERTERS,
PARALLEL LINE INTERFACES,
SERIAL LINE INTERFACES,
DMA INTERFACES,
BUS ACCESSORIES, ETC.

Typical LSI-11 System Configuration

General Registers

The LSI-11 central processor module contains eight 16-bit general-purpose registers that can perform a variety of functions. These registers can serve as accumulators, index registers, autoincrement registers, autodecrement registers, or as stack pointers for temporary storage of data. Arithmetic operations can be from one general register to another, from one memory location or device register to another, or between memory locations or a device register and a general register. The eight 16-bit general registers (R0 through R7) are identified in the following figure.



General Register Identification

Registers R6 and R7 in the LSI-11 are dedicated. R6 serves as the Stack Pointer (SP) and contains the location (address) of the last entry in the stack. Register R7 serves as the processor Program Counter (PC) and contains the address of the next instruction to be executed. It is normally used for addressing purposes only and not as an accumulator. Register operations are internal to the processor and do not require bus cycles (except for instruction fetch); all memory and peripheral device data transfers do require bus cycles and longer execution time. Thus, general registers used for processor operations result in faster execution times. The bus cycles required for memory and device references are described below.

Bus Cycles

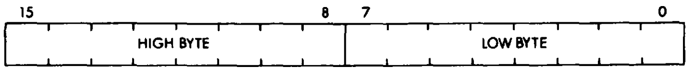
The bus cycles (with respect to the processor) are:

DATI	Data word transfer input	Equivalent to Read operation
DATIO	Data word transfer input followed by word transfer output	Equivalent to Read/Modify Write

DATIOB	Data word transfer input followed by byte transfer output	Equivalent to Read/Modify Write

Addressing Memory and Peripherals
The maximum direct address space of the LSI-11 is 64K bytes. LSI-11 memory locations and peripherals device registers are addressed in the same manner. The upper 4096 addresses (56K-64K bytes) are reserved (by PDP-11 convention) for peripheral device addressing.

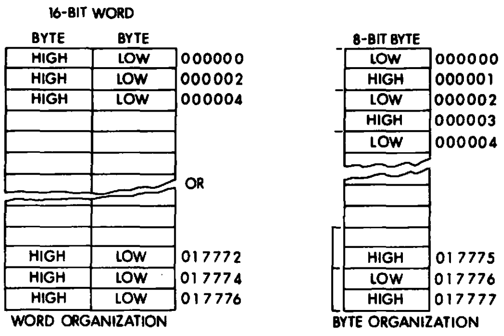
An LSI-11 word is divided into high byte and a low byte as shown in this figure.



High and Low Byte

Word addresses are always even-numbered. Byte addresses can be either even- or odd-numbered. Low bytes are stored at even-numbered memory locations and high bytes at odd-numbered memory locations. Thus, it is convenient to view the memory as shown in the accompanying figure.

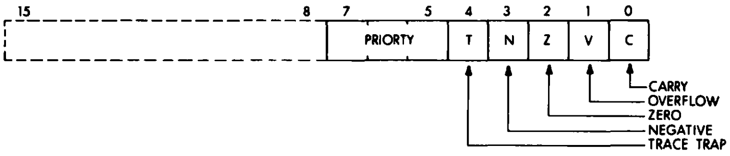
Certain memory locations have been reserved by convention for interrupt and trap handling and peripheral device registers. Addresses from 0 to 376_8 are reserved for trap and device interrupt vector locations. Several of these are reserved in particular for system (processor initiated) traps.



Word and Byte Addresses for First 4K Bank

Processor Status Word

The Processor Status Word (PS) contains information on the current processor status. This information includes the current processor priority, the condition codes describing the arithmetic or logic results of the last instruction, and an indicator for detecting the execution of an instruction to be trapped during program debugging. The PS word format is shown in the figure. Certain instructions allow programmed manipulation of condition code bits and loading or storing (moving) the PS.



Processor Status Word (PS)

Interrupt Priority Bit

The processor operates with interrupt priority PS bit 7 asserted (1) or cleared (0). When PS bit 7 = 1, an external device cannot interrupt the processor with a request for service. The processor must be operating at PS bit 7 = 0 for the device request to be serviced by the processor. As compared to other PDP-11s, the LSI-11 and LSI-11/2 service interrupts at one priority level. The LSI-11/23 services interrupts at four levels of priority.

Condition Codes

The condition codes contain information on the result of the last CPU operation. The bits are set as follows (the bits are set after execution of arithmetic or logical, single operand or double operand instructions):

Z = 1,	If the result was zero.
N = 1,	If the result was negative.
C = 1,	If the operation resulted in a carry from the MSB (most significant bit) or a 1 was shifted from the MSB or LSB (least significant bit).
V = 1,	If the operation resulted in an arithmetic overflow.

Trap (T Bit)

The program can only set or clear the trap bit (T) by popping a new PS off the stack. When set, a processor trap will occur through location 14 at completion of the current instruction execution, and a new processor status word will be loaded from location 16. This T bit is especially useful in debugging programs since it allows programs to single-step instructions.

Instruction Set

The PDP-11 instruction set permits you to take advantage of Digital Equipment Corporation's years of experience with the PDP-11 family. Available are: application notes, software, documentation, training, and the DECUS user library of application programs.

The instruction set uses the flexibility of the general-purpose registers to provide more than 400 instructions—the most comprehensive and powerful instruction set of any 16-bit computer. Unlike 16-bit computers which have three classes of instructions (memory reference instructions, operate or accumulator control instructions, and I/O instructions), all data manipulation operations in the LSI-11 are accomplished with one set of instructions. Instructions that are used to manipulate memory locations can be used with peripheral device registers. For example, data in an external device register can be tested or modified directly without bringing it into memory or disturbing the general registers. Programs can add or compare data either logically or arithmetically in a device register.

LSI-11 instructions use both single and double operand addresses for words or bytes. The LSI-11, therefore, performs in one step such operations as adding or subtracting two operands, or moving an operand from one location to another.

LSI-11 Approach

ADD A, B	Add contents of location A to location B; store results at location B

Conventional Approach

LDA A	Load contents of memory location A into accumulator
ADD B	Add contents of memory location B to accumulator
STA B	Store result at location B

Addressing

Much of the power of the LSI-11 is derived from its wide range of addressing capabilities. LSI-11 addressing modes include sequential forward or backward addressing, address indexing, indirect addressing, 16-bit word addressing, 8-bit byte addressing, and stack addressing. Variable-length instruction formatting allows a minimum number of words to be used for each addressing mode. The result is efficient use of program storage space.

LSI-11 MEMORY ORGANIZATON

LSI-11 memory organization is shown in the accompanying figure.

THE LSI-11 BUS

The LSI-11 bus is a simple, fast, easy-to-use interface between LSI-11 modules. All LSI-11 modules connected to this common bidirectional bus structure receive the same interface signal lines. A detailed description of the LSI-11 bus is included in Chapter 8.

Bus data and control lines (except daisy-chain grant signals) are bidirectional open-collector lines that are asserted low. The LSI-11 processor uses 16 data/address lines and 17 control synchronization and system function lines.

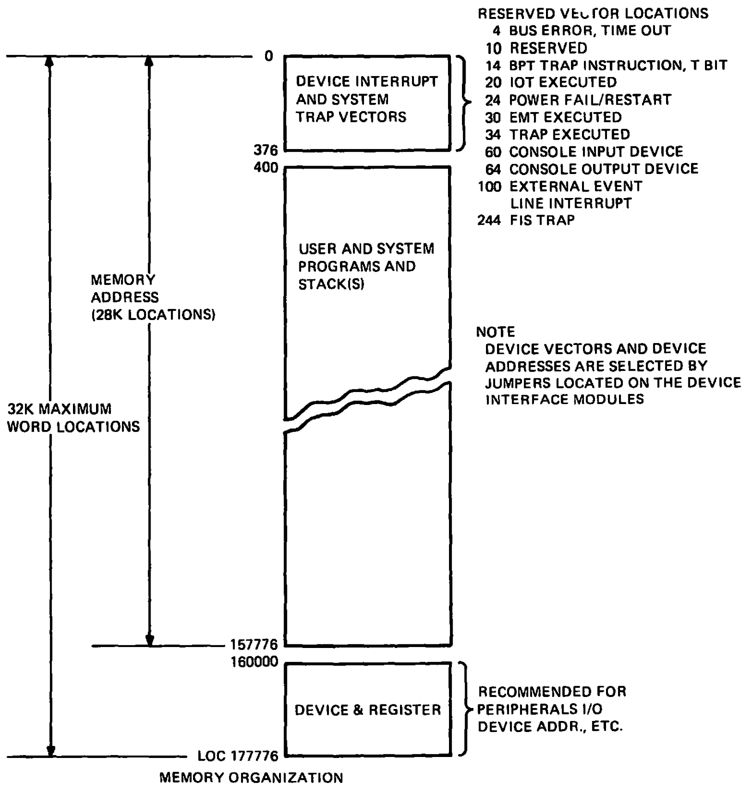
NOTE

The LSI-11 bus has recently been expanded to accommodate 18-bit addresses, thus increasing maximum system memory size to 248K bytes. More detailed information is available in Chapter 8: LSI-11 Bus and Chapter 9: Memory Management.

Both 16-bit address and 8-bit bytes or 16-bit data words are multiplexed over 16 data/address lines. During a programmed data transfer, the processor will assert an address on the bus for a fixed time. After the address time has been completed, the processor initiates the programmed input or output data transfer. The actual data transfer is asynchronous and requires a reply from the addressed device; bus synchronization and control signals provide this function.

With bidirectional and asynchronous communications on the LSI-11 bus, devices can send, receive, and exchange data at their own rates. The bidirectional nature of the bus allows utilization of common bus interfaces for different devices and simplifies the interface design.

Communication between two devices on the bus is in the form of a master-slave relationship. At any point in 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". A typical example of the relationship is the processor, as master, fetching an instruction from memory (which is always a slave). Another example is a DMA device interface, as master, transferring data to memory, as slave. Bus master control is dynamic. The bus arbitrator is the processor module; it may pass bus control to a DMA device. The DMA device, as bus master, could then communicate with memory (always a slave) without processor intervention.


RESERVED VECTOR LOCATIONS
4 BUS ERROR, TIME OUT
10 RESERVED
14 BPT TRAP INSTRUCTION, T BIT
20 IOT EXECUTED
24 POWER FAIL/RESTART
30 EMT EXECUTED
34 TRAP EXECUTED
60 CONSOLE INPUT DEVICE
64 CONSOLE OUTPUT DEVICE
100 EXTERNAL EVENT LINE INTERRUPT
244 FIS TRAP
0
DEVICE INTERRUPT AND SYSTEM TRAP VECTORS
376
400
USER AND SYSTEM PROGRAMS AND STACK(S)
MEMORY ADDRESS (28K LOCATIONS)
32K MAXIMUM WORD LOCATIONS
NOTE
DEVICE VECTORS AND DEVICE ADDRESSES ARE SELECTED BY JUMPERS LOCATED ON THE DEVICE INTERFACE MODULES
157776
160000
DEVICE & REGISTER
RECOMMENDED FOR PERIPHERALS I/O DEVICE ADDR., ETC.
LOC 177776
MEMORY ORGANIZATION

NOTE
THERE IS 32K OF USERS MEMORY SPACE AVAILABLE; HOWEVER 0–28K IS RECOMMENDED FOR MEMORY ADDRESS LOCATIONS, AND 28K–32K FOR PERIPHERALS I/O DEVICE ADDRESSES, ETC.

Memory Organization

Since the LSI-11 Bus is used by the processor and all I/O devices, a hardware priority structure determines which device becomes bus master when more than one device requests control of the bus. Every device on the LSI-11 Bus which is capable of becoming bus master is assigned a priority according to its electrical position on the bus. The device closet to the processor has the highest priority.

Data transfers on the LSI-11 Bus are asynchronous so that communication is independent of the physical bus length and the response time of the slave device. The asynchronous operation between bus master and slave devices precludes the need for synchronizing bus transactions with clock signals. Thus, each device is allowed to operate at the maximum possible speed.

Full 16-bit words or 8-bit bytes of information can be transferred on the bus between a master and a slave. The information can be instructions, addresses, or data. This type of information transfer occurs when the processor, as master, is fetching instructions, operands, and data from memory, and is storing the results in memory after execution of the instruction.

Information is provided in this handbook about the assembly language parameters, processes, and techniques involved in programming the LSI-11. DIGITAL publishes tutorial software documentation that provides detailed information about using the LSI-11 instruction set to develop programs. There are also well-developed courses for customers by DIGITAL's Education Services group.

The material presented on the LSI-11 instruction set, addressing modes, and programming techiques is intended, with the examples included, to illustrate the range of and possibilities for program development.

PERIPHERALS

DIGITAL manufactures a full range of peripheral equipment designed to meet specific needs as well as to maintain 11-family compatibility. I/O and storage devices range from paper tape readers through high volume disk packs and from the DECwriter to the intelligent terminals which provide both hard copy and video display. There is a complete spectrum of peripheral devices available to complement the software, and to provide the complete answer to customer needs in all product line areas — business, education, industry, laboratory, and medicine.

The *Microcomputer Interfaces Handbook* describes in detail the optional equipment available for use with the LSI-11 microcomputers.

DOCUMENTATION

DIGITAL offers several levels of technical documentation describing 11-family software and hardware. The Microcomputer Handbook series presents an introductory technical level of LSI-11 family information. The hardware user documentation, which accompanies the delivery of LSI-11 computer systems and peripherals, offers the most detailed levels of information. There are also several good books put out by commercial publishers which discuss the LSI-11 family. Specific topics, such as microprogramming, are also well-covered in commercially available books. If you have a specific documentation need, discuss the issue with a DIGITAL salesperson, who will guide you to the appropriate literature.

NUMERICAL NOTATION

Three number systems are used in this handbook: octal, base eight; binary, base two; and decimal, base ten. **Octal** is used for address locations, contents of addresses, and instruction operation codes. **Binary** is used for descriptions of words and **decimal** for normal quantitative references.

EDUCATIONAL SERVICES

Like DIGITAL's computer systems, training facilities span the globe—Japan, Australia, Great Britain, Germany, France, the Netherlands, Sweden, Italy, Canada and throughout the United States. Services are centered around 14 fully equipped Regional Education Centers and a staff of seasoned educators dedicated to providing all aspects of education and training needed in support of all DIGITAL systems.

Catalog courses are regularly scheduled classes offered at training centers. Presently there are more than 100 scheduled classes that cover the range from first-time user to highly specialized training on theory of operation. Most catalog courses include extensive hands-on laboratory time, and all incorporate the use of a broad assembly of student workbooks, reference manuals, and other instructional materials.

Specialized training is available for users with unique applications or training situations. This approach is designed to give the student the maximum relevant material for specific applications, while minimizing extraneous information. The custom courses are tailored to the individual customer's schedule and typically comprise a series of courses. These can be modified from existing courses or be entirely new programs based on mutually agreed upon objectives.

Customers with a group of individuals to train may find it more economical to have Educational Services conduct courses at the user's home site. On-site instruction of both catalog and custom courses eliminates travel and other expenses incurred by students attending classes at training centers. This method of instruction further enhances training by allowing DIGITAL instructors to emphasize points of particular value to the student's applications and operations.

By taking advantage of the latest in audio-visual techniques, Educational Services has developed a series of courses that offers independent learning. Audio-visual courses are convenient, self-contained, and modular in topic. The self-instructional format allows student to progress at ther own rates, study when and where they wish, and play back modules for review. Audio-visual course material is available in several forms—video-tape, videocassette, or audio/filmstrip cassette—all supported by student workbooks.

LSI-11 RELATED COURSES

DIGITAL's educational group offers a series of courses on the hardware and software of your LSI-11 systems. For complete information on course content, prerequisites, pricing, and scheduling, consult the Educational Courses Catalog available through DIGITAL's Educational Centers listed below:

Boston area:
Digital Equipment Corporation
Educational Services Department
12 Crosby Drive
Bedford, MA 01730

Chicago area:
Digital Equipment Corporation
Educational Services Department
5600 Apollo Drive
Rolling Meadows, Illinois 60008
Telephone: (312) 640-5520

Philadelphia area:
Digital Equipment Corporation
Educational Services Department
Whitpain Office Campus
1740 Walton Road
Blue Bell, Pennsylvania 19422
Telephone: (215) 825-4200 Ext. 26

Washington, D.C. area:
Digital Equipment Corporation
Education Services Department
Lanham 30 Office Building
5900 Princess Garden Parkway
Lanham, Maryland 20801
Telephone: (301) 459-7900 Ext.315 or 215

San Francisco area:
Digital Equipment Corporation
Educational Services Department
310 Soquel Way
Sunnyvale, Calfornia 94086
Telephone: (408) 984-0200 Ext. 293 or 294

DECUS

Additional programs and applications packages may be obtained from DECUS, the Digital Equipment Computer Users Society. DECUS is a not-for-profit computer users group (the largest such group, worldwide) that sponors technical symposia, publishes a periodic newsletter and symposia proceedings, and maintains a large library of programs for the various DIGITAL computers. Every customer who has purchased or ordered a computer manufactured by DIGITAL is eligible for an installation membership in DECUS. Associate membership is also available to any person with a bona fide interest in DIGITAL computers. Membership in DECUS is strictly voluntary, and does not require payment of dues. Programs from the DECUS library are available to all members for nominal reproduction and handling charges. A complete catalog of available programs may be obtained from the society.

Further information on the DECUS Library, publications, and other DECUS activities is available from the DECUS offices listed below:

AUSTRALIA AND NEW ZEALAND:
DECUS
P.O. Box 491
Crows Nest, N.S.W. 2065
Australia

EUROPE AND MIDDLE EAST:
DECUS
Case Postale 340
1211 Geneva 26
Switzerland

CANADA:
DECUS
P.O. Box 11500
Ottawa, Ontario K2H 8K8
Canada

ALL OTHERS:
DECUS
146 Main Street
Maynard, Massachusetts 01754
U.S.A

MAINTENANCE

DIGITAL offers a wide range of maintenance services to LSI-11, PDP-11/03, and PDP-11 customers. These services are provided through DIGITAL's Customer Services Organization and have been designed to meet our customer's complete maintenance needs, either on-site or off-site. These service plans provide complete DIGITAL maintenance on-site by our factory-trained engineers, or provide module and unit repairs off-site for those customers desiring to perform their own maintenance.

ON-SITE SERVICE

DIGITAL's service organization provides on-site maintenance service with a staff of over 5,800 factory-trained engineers in 360 locations world-wide. Each service office maintains adequate inventory to support its customers and is fully supported by our logistics operation in Maynard, Massachusetts.

- Service Agreement — On-site contract service is available for all LSI-11 based systems, subject to minimum hardware configurations. This service provides corrective maintenance, preventive maintenance, and all applicable engineering changes to ensure your products are operational and kept completely up to date. In addition to priority service, contractual maintenance allows DIGITAL customers to budget for their annual maintenance needs. The monthly contract charge covers all travel, labor, and material.

- Per Call — DIGITAL also offers on-site per call service. DIGITAL will respond to maintenance needs on a billable travel, time, and materials basis.
- Installation and Warranty — On-site installation and warranty-service is also available for LSI-11 based systems, subject to minimum hardware configurations. This service must be purchased at the time of original order.

Off-Site Service
DIGITAL offers complete unit and module repair services to customers capable of performing their own maintenance. The Customers Returns Area (CRA) has been established in Woburn, Massachusetts, to offer single-point interfacing for all off-site repairs for North American customers. The CRA assures the customer of complete "one-step shopping" for all factory-level warranty and post-warranty services. All repairs are effected at our Module Repair Facility in Woburn.

For Eurpean, Australian, and Japanese customers, we have established Product Repair Centers (PRCs) in eleven countries. Customers can return defective material to the PRC in their country without the burden of customs, duties, and licensing requirements. The PRCs offer the same services to these customers as the CRA in Woburn.

For information on services in Latin and South America, contact the CRA in Woburn.

- **Warranty Service**—All products are warranted against defects in workmanship and materials under normal proper use in their unmodified condition for a period of ninety (90) days from date of initial shipment. As a condition of this warranty, customers must obtain a DIGITAL Repair Authorization (RA) number and return the products prepaid, together with a written description of the claimed defect, to the nearest authorized DIGITAL Repair Center as listed here.

  RA numbers may be obtained by contacting the CRA in Woburn (PRC if non-U.S.) and providing the following information:

  1. Customer name and location.
  2. Part number/serial number of failing item.

3. Part number/serial number of next higher assembly if a module or subassembly.
4. Product line and date purchased.

- **Post-Warranty Service**—DIGITAL offers its post-warranty services in several forms:
  1. **Loose piece subassembly repair,** for a minimum order.
  2. **Prepaid module mailers.** Available on specific module types.
  3. **Firm quote product and option repair.** For the smaller customer with only occasional service needs, and those who do not have any in-house troubleshooting capability.

For more complete information and pricing on any of the services listed, contact the repair center nearest you.

The following repair centers have been established to provide complete off-site repair services. These centers should be contacted for all off-site warranty and post-warranty services and prices. All defective material should be sent to the address indicated with your RA number appearing on the shipping label.

North America
Digital Equipment Corporation
Customer Returns Area
36 Cabot Road
Woburn, MA 01801

RA Number
Telephone Number: 617-933-8710

Canada
Digital Equipment of Canada, Ltd.
100 Herzberg Road
Kanata, Ontario, Canada

RA Number
Telephone Number: 613-592-5111

Europe

Belgium
Product Repair Center Manager
Digital Equipment Sa/Nv
Brand Whitlock Boulevard 87
B-1040 Bruxelles, Belguim
Telephone: (02) 733-9650

France
Product Repair Center Manager
Digital Equipment France
7, Rue de L'Esterel Silic 225
94528 Rungis, Cedex, France
Telephone: (01) 687-2333

Germany
Product Repair Center Manager
Digital Equipment GmbH
D-8000 Munchen 40
Wallensteinplatz 2
West Germany
Telephone: (89) 35031

Holland
Product Repair Center Manager
Digital Equipment Bv
Coloradodreef 26-28
P. O. Box 9064
3563 Utrecht, Holland
Telephone: (030) 61 1814

Italy
Product Repair Center Manager
Digital Equipment S.P.A.
Viale Fulvio Testi 117
Cinisello Balsamo
20092 Milan, Italy
Telephone: (02) 61797

Sweden
Product Repair Center Manager
Digital Equipment AB
Box 1250
S-17124 Solna 1
Sweden
Telephone: (08) 730-08-00

Switzerland
Product Repair Center Manager
Digital Equipment Corp. AG/SA
Koeschenruetistr 116
CH-8052 Zurich
Switzerland
Telephone: (01) 5152 66

United Kingdom
Product Repair Center Manager
Digital Equipment Corp., Ltd.
Gasworks Road
Building V.7.A.B.L. Site
Reading RGI-3EF
England
Telephone: (734) 58 35 55

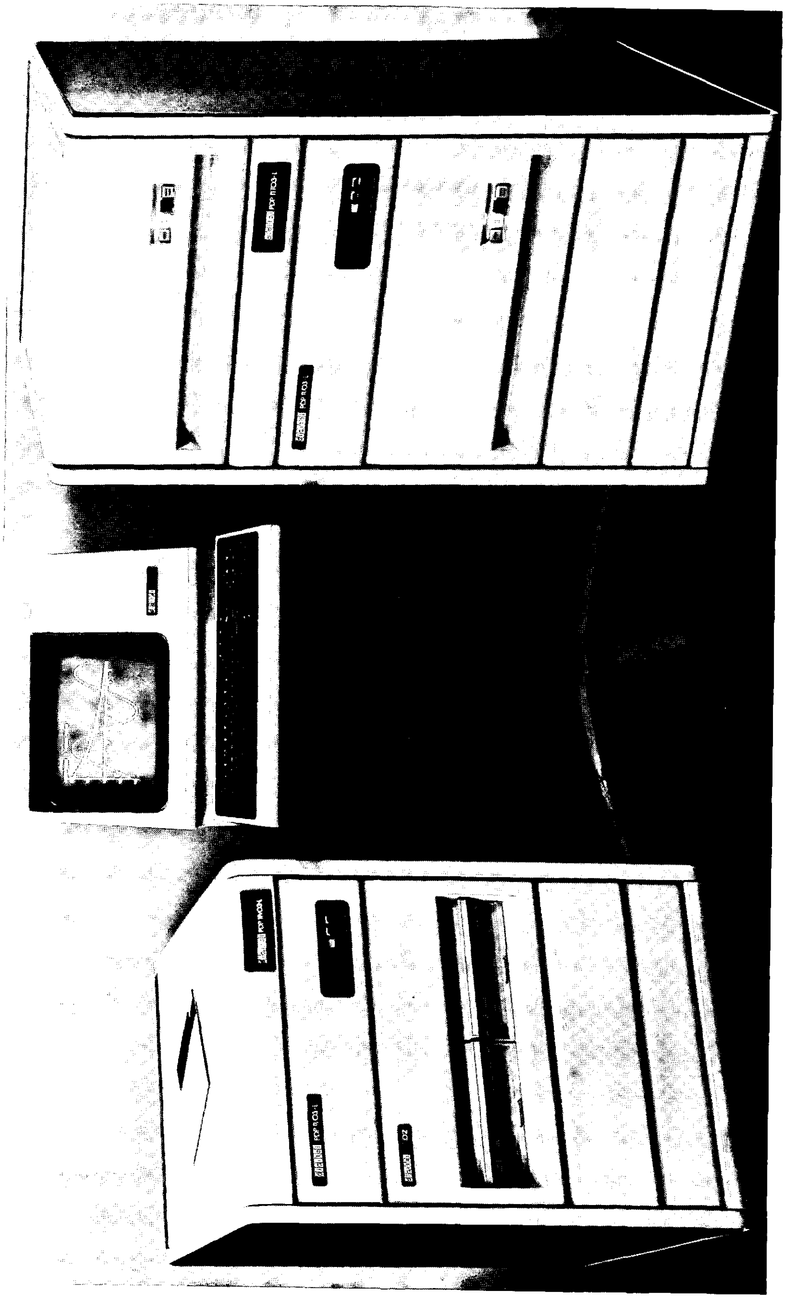
General International Area
At this time, the only services offered in the GIA is firm quote product/option and loose piece subassembly repairs through the Tokyo and Sydney repair centers.

GIA Product Repair Centers

Australia
Product Repair Center Manager
Digital Equip. Autralia Pty. Ltd.
132-125 Willoughby Road
P.O. Box 491
Crows Nest
New South Wales, 2065 Australia
Telephone: (02) 439-3598

Latin America
South America
Contact the CRA, Woburn

Japan
Product Repair Center
Digital Equipment Corp. Int.
#1, Taiso Shinjuku Bldg.
1-26-12, Shinjuku/K.U.
Tokyo 160, Japan
Telephone: (3) 341 5481



CHAPTER 2 SYSTEMS AND SOFTWARE

DIGITAL's LSI-11 processors are named as members of the PDP-11 family when they are combined with boxes, power supplies, and backplanes.

The LSI-11 and LSI-11/2 processors, when configured as part of a boxed computer, are known as PDP-11/03 or PDP-11/03L **boxes,** depending upon the box. When the boxes are configured into packaged systems, they are called PDP-11V03L or PDP-11T03L **systems**.

The newest member of the LSI-11 family, the LSI-11/23, when configured in a box, is called a PDP-11/23 and in a system configuration is called a PDP-11V23 or PDP-11T23.

DIGITAL makes this family of low-end computers available at several levels of integration allowing you maximum design flexibility.

DIGITAL's family of PDP-11 peripherals includes an extensive line of hardware interfaces that may be used with the LSI bus based systems. These are described in detail in the *Microcomputer Interfaces Handbook.*

PDP-11/23 SYSTEM DESIGNATIONS

Box Option Designations

The PDP-11/23 box options include: the processor, two or four memory boards (for 128K or 256K bytes), the BA11-N box, and the DLV11 interface.

- 11/23-AA = BA11-NC, KDF11-AA, (2) MSV11-DD, BDV11-AA, DLV11-J, 120V, 50/60 Hz, 128 KB
- 11/23-AB = BA11-ND, KDF11-AA, (2) MSV11-DD, BDV11-AA, DLV11-J, 240V, 50/60 Hz, 128 KB
- 11/23-AC = BA11-NC, KDF11-AA, (4) MSV11-DD, BDV11-AA, DLV11-J, 120V, 50/60 Hz, 256 KB
- 11/23-AD = BA11-ND, KDF11-AA, (4) MSV11-DD, BDV11-AA, DLV11-J, 240V, 50/60 Hz, 256 KB

RL Systems Option Designations

The RL systems include the earlier listed box systems with the RLV11 interface and two RL01s—5.24M bytes of data storage per RL01 (hard) disk drive.

- 11T23-AA = 11/23-AA, RLV11, (2) RL01-AK, H9612-AC cabinet, 871-A power controller, 120V, 50/60 Hz, 12A, 128 Kb
- 11T23-AB = 11/23-AB, RLV11, (2) RL01-AK, H9612-AC cabinet, 871-B power controller, 240V, 50/60 Hz, 8A, 128 Kb
- 11T23-AC = 11/23-AA, RLV11, (2) RL01-AK, H9612-AC cabinet, 871-C power controller, 120V, 50/60 Hz, 16A, 128 Kb

RX Systems Option Designations

The RX systems include the earlier listed boxed systems with the RXV21 interface and two RX02s (floppy drives) with 512K bytes of data storage per drive.

- 11V23-AA = 11/23-AA, RXV21, RX02-BA, H9610-AC cabinet, 871-A power controller, 120V, 60 Hz, 12A, 128 Kb
- 11V23-AC = 11/23-AA, RXV21, RX02-BC, H9610-AC cabinet, 871-A power controller, 120V, 50 Hz, 12A, 128 Kb
- 11V23-AD = 11/23-AB, RXV21, RX02-BD, H9610-AC cabinet, 871-B power controller, 240V, 50 Hz, 8A, 128 Kb

BA11-N MOUNTING BOX

The PDP-11/23 boxes and systems utilize the BA11-N box. For more specific details on the box, see the *Microcomputer Interfaces Handbook*.

SPECIFICATIONS

Input Power

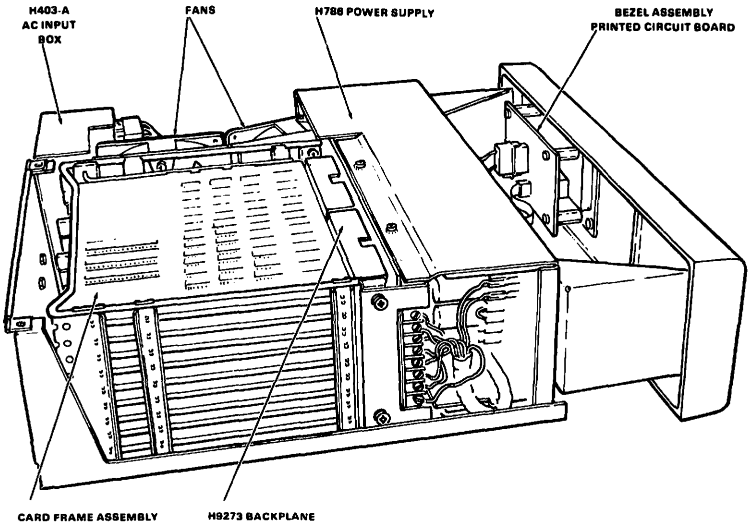
Voltage:	100-126 Vrms or 200-254 Vrms (switch-selected)
Frequency:	48-63 Hz
Power:	1380 W (maximum) including convenience outlet 633 W (maximum) power supply and modules

Environmental

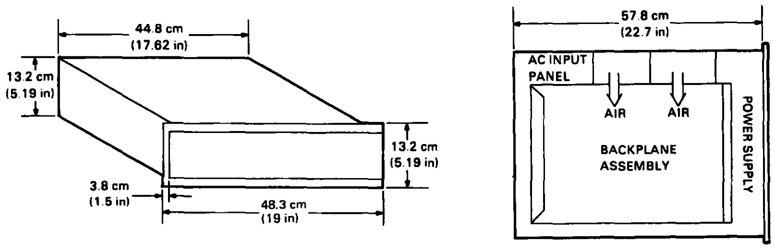
Temperature:	5° - 40° C (41° - 104° F) Derate at 6° C (11° F)/1000 ft at altitudes above 8000 ft
Relative Humidity:	10% to 95% (no condensation)

Mechanical

Height:	13.2 cm (5.19 in)
Width:	48.3 cm (19 in)
Depth:	without mounting brackets - 57.8 cm (22.75 in) with mounting brackets - 67.98 cm (26.76 in)



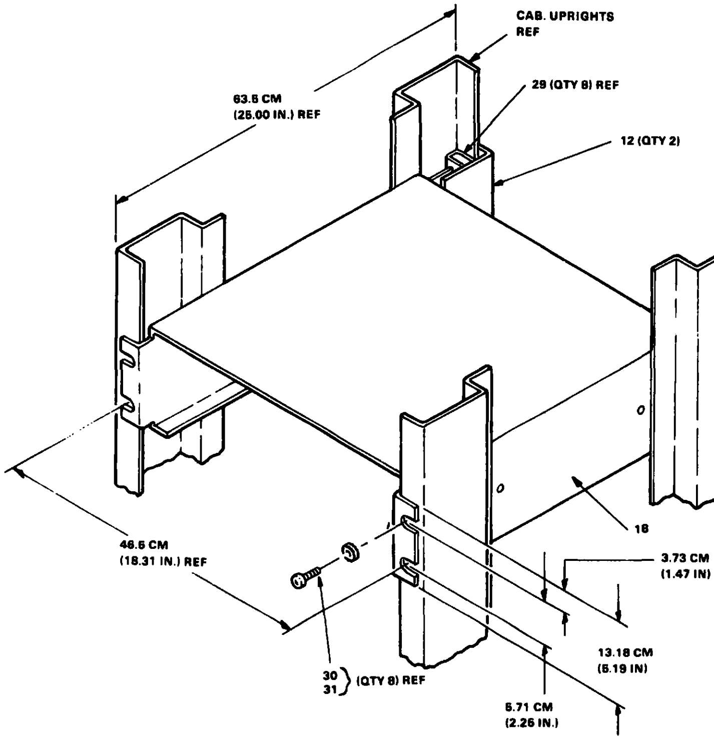
BA11-N Major Assemblies



BA11-N Assembly Unit

SOFTWARE

The RL, hard disk based, PDP-11/23 system is offered with two operating systems, **RT-11** and **RSX-11**. Both support the PDP-11/23 memory management feature; however, the design and implementation of this support varies between the two systems.



BA11-N Cover Mounting Dimensions

The RT-11 XM monitor feature can be utilized by a PDP-11/23 when the system is used by a single user requiring no more that two concurrent tasks. The XM monitor requires the user to handle program mapping above 56 Kb. It is, however, very efficient in supporting large programs where data exists only above 56 Kb. The present release of RT-11 (V3.B) is capable of supporting the PDP-11/23.

RXS-11M Features:
- multiuser
- multitasking
- checkpointing
- file protection
- mass storage base
- file storage and retrieval
- device independence

RSX-11M supports a variety of high-level languages, including FORTRAN IV, FORTRAN IV-PLUS, BASIC, BASIC PLUS-2, MACRO, CORAL 66, COGO and ANSI 74 COBOL. It also enables a number of programmers to share the 11/23 simultaneously while developing and debugging their programs.

Data management capability is provided by DATATRIEVE, DIGITAL's special inquiry and report writing language, by RMS-11 record management system, and by SORT-11. For intersystem communication, DIGITAL offers 2780/3271 protocol emulation and the industry's most advanced networking system, DECnet-11M.

RSX-11S Features:
- memory-based
- multitasking
- subset of RSX-11M

As the smallest member of the RSX-11 family of real-time, multitasking operating systems, RSX-11S provides a dedicated, execute-only environment for monitoring and controlling many real-time processes concurrently. It is implemented as a memory-based, compatible subset of RSX-11M and, thus, is not dependent on any mass storage device for execution. RSX-11S system generation and program development take place on a host RSX-11M system. It supports non-file-structured data storage, runs programs written in FORTRAN and MACRO, and can be connected to other systems using DECnet-11S.

RT-11 Features:
- real-time
- single-job or foreground/background program execution
- supports 256 Kb of main memory
- mass storage-based
- easy-to-use command languages
- powerful editor
- debugging facilities

RT-11 is DIGITAL's single-user, foreground/background operating system. Fully interrupt-driven, overlapped input/output provide fast program execution and low overhead. Simultaneous execution of foreground/background tasks optimizes system capability. And a nested "common file" execution allows frequently used groups of system commands to be stored and recalled for execution with one simple command.

Like RSX-11M, RT-11 supports a variety of high-level programming languages, including BASIC and Multiuser BASIC, FORTRAN IV, MACRO, APL, and FOCAL, DIGITAL's high-level interactive programming language for data acquisition and analysis. Data management capability is provided by FMS-11, DIGITAL's file management service. Communication with other systems can be effected through 2780 protocol emulation, DECnet-RT, and RT^2/PDT, software that allows program loading from an RT-11 based system directly into our PDT-11/150 intelligent terminals.

CTS-300 Features:

- timesharing
- fast terminal response
- easy to use
- interactive
- full-service utility routines
- line printer spooling
- dynamic memory allocation
- intertask communications
- multivolume files
- file sharing

CTS-300 (DIGITAL's Commercial Transaction Processing System) is a disk-resident, business-oriented operating system. Using our COBOL-like DIBOL-11 programming language, it provides simultaneous multiuser, multiterminal capability for transaction processing. Depending on the application program size, CTS-300 can support up to eight concurrent tasks. System access is gained through VT100 high-speed video terminals. Fast terminal response and greater memory availability are achieved because programs reside in memory in dynamic partitions while running.

Data management capability is provided by DIGITAL's DECFORM, DMS-300, SORT/MERGE, and ISMUTL (which creates and maintains ISAM files). Intersystem communication is accomplished via DICAM or 3271/2780 protocol emulation.

PDP-11/03-S AND PDP-11/03-L BOXES

The PDP-11/03-S and -L are boxed versions of the LSI-11 microcomputer. They includes a rack-mountable enclosure containing the LSI-11 processor, memory, and LSI bus-structured backplane, a power supply for the processor and memory, EIS/FIS chip standard, and use the double height processor and memory modules.

PDP-11/03-S (SMALL BOXES)

The 11/03-S Small Boxes (4 X 4 backplane)

Option #	Description
11/03-SC	(KD11-HA, MSV11-DC, KEV11, BA11-MA)
11/03-SD	(KD11-HA, MSV11-DC, KEV11, BA11-MB)
111/03-SE	(KD11-HA, MSV11-DD, KEV11, BA11-MA)
11/03-SF	(KD11-HA, MSV11-DD, KEV11, BA11-MB)

BA11-M MOUNTING BOX

The PDP-11/03-S includes the BA11-M box. For specific information on the BA11-M box, see the *Microcomputer Interfaces Handbook.*

SPECIFICATIONS

Environmental

Temperature:	5°-40° C (41°-104° F) Derate at 6° C (11°F)/1000 ft at altitudes above 8,000 ft.
Relative Humidity:	10% to 95% (no condensation)

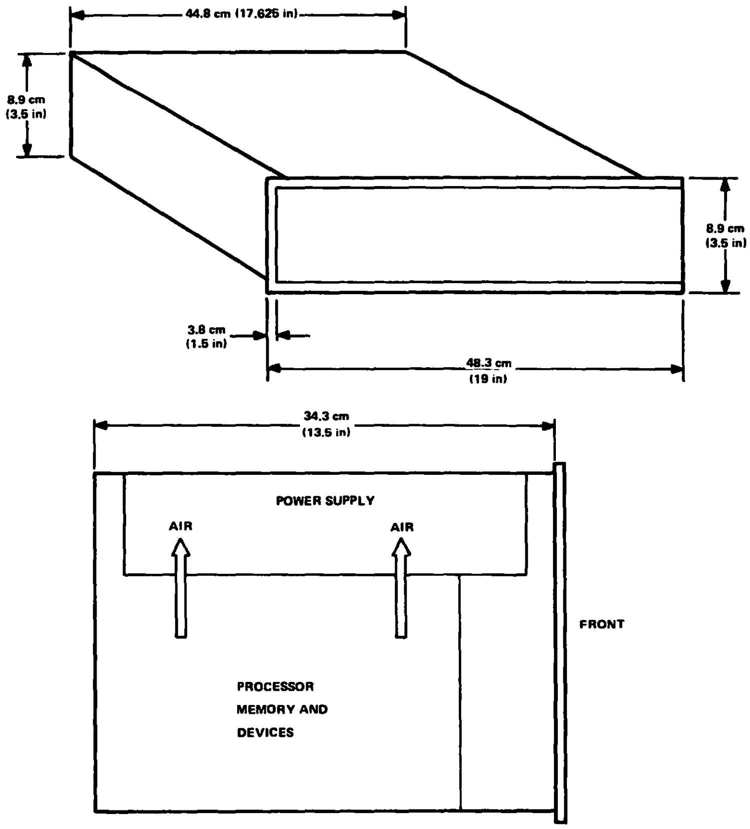
PDP-11/03-L BOX

The PDP-11/03-L is a packaged and boxed version of the LSI-11 microcomputer. It includes a rack-mountable enclosure, the BA11-N box containing the LSI-11 processor and 32K bytes of memory, an LSI-11 bus-structured backplane, a bootstrap/diagnostic module, a 240 watt power supply, and a switch/indicator panel.

This box system retains the features of the smaller PDP-11/03s; however, the H786 power supply is capable of producing more than twice the dc power available in the smaller system (+5 V, 22 A and +12 V, 11 A). The H9273 backplane also has the added feature of having an interconnect scheme on the -CD side of the bus to let users configure special-purpose system functions.

The PDP-11/03-L box is available in four models, depending upon the memory and line cord supplied. The H786 power supply is universal; it can operate on 115 or 230 Vac, 50 or 60 Hz primary power. A switch is provided on the back of the BA11-N box for this purpose.

Model	Description
11/03-LH	KD11-HA, MSV11-DC, KEV11, BDV11-AA, BA11-NC
11/03-LJ	KD11-HA, MSV11-DC, KEV11, BDV11-AA, BA11-ND
11/03-LK	KD11-HA, MSV11-DD, KEV11, BDV11-AA, BA11-NC
11/03-LL	KD11-HA, MSV11-DD, KEV11, BDV11-AA, BA11-ND



BA11-M Assembly Unit

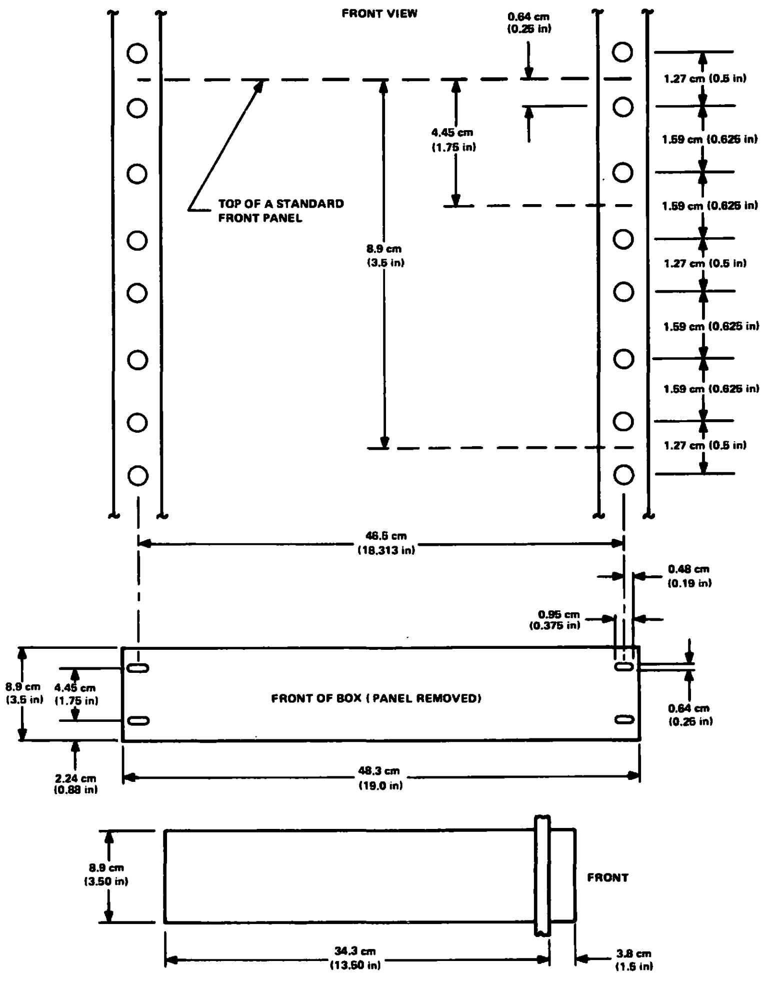
Packaged Systems are:

SR-VXSSA	11/03-LH, LJ
SR-VXSSB	11/03-LK, LL
SR-VXLLB	11/03-LK, LL

PDP-11/03-L Specifications

Input Power

Voltage:	100-127 Vrms or 200-254 Vrms (switch-selected)
Frequency:	48-63 Hz



BA11-M Cabinet Mounting

Power: 1380 W (maximum) including convenience outlet
633 W (maximum) power supply and modules

Environmental

Temperature:	5°-40° C (41°-104° F) Derate at 6° C (11°F)/1000 ft at altitudes above 8000 ft.
Relative Humidity:	10% to 95% (no condensation)

Mechanical

Height:	13.2 cm (5.19 in.)
Width:	48.3 cm (19 in.)
Depth:	without mounting brackets—57.8 cm (22.75 in.) with mounting brackets—67.96 cm (26.75 in.)
Mounting Dimensions:	Refer to the information on the BA11-N box in the PDP-11/23 section.

PDP-11V03-L PACKAGED SYSTEMS

The PDP-11V03-L is a complete, floppy disk-based packaged system that includes all necessary hardware, factory-configured and installed. The dual-drive, floppy disk system stores 1.0M bytes and is mounted in an attractive caster-equipped cabinet. The system is available in several variations, depending upon choice of terminal, input power, and the software operating system. The RT-11 single-user disk operating system is a selectable item in the packaged systems. Other compatible PDP-11 products, such as BASIC, FORTRAN or APL, are available as optional add-ons.

All PDP-11V03-L systems include the PDP-11/03-L microcomputer, which consists of the processor with KEV11 EIS/FIS extended arithmetic capability, 32K byte or 64K byte MOS memory, one or four serial line interfaces, BDV11 bootstrap/loader, and the RXV21 dual floppy disk system.

PDP-11V03-L Specifications Summary

Physical

	Height	**Depth**	**Width**
H9610-AA, AB Cabinet	75.3 cm (30.9 in)	75.3 cm (30.9 in)	53.3 cm (21.25)

Electrical

Input Voltage	104-126 Vac or 209-259 Vac
Frequency	50 Hz ±0.5 Hz or 60 Hz ± 0.5 Hz

Environmental

Power Consumption:	Typical (or idle)	Maximum
Computer Cabinet	(115 V) 1100 W	1240 W
	(230 V) 1120 W	1265 W

PDP-11T03-L SYSTEMS

The PDP-11T03-L is a hard disk system that contains two top-loading RL01 cartridge disk drives providing a total of 10.4M bytes of storage. The system is available in several variations, depending upon choice of terminal, input power, and software. The RT-11 operating system software is available as part of the standard packaged systems. Other compatible PDP-11 software products, such as BASIC, FORTRAN, or APL, are available as add-on options.

All PDP-11T03-L systems include the processor with EIS/FIS extended arithmetic capability, 64K byte MOS memory, DLV11-J four-line serial interface, BDV11 bootstrap/loader, and the RLV11 dual disk drive subsystem.

PDP-11T03-L Specifications Summary

Physical

	Height	**Depth**	**Width**
H9612-AA, -AB Cabinet	101.7 cm (40.5 in)	75.3 cm (30 in)	53.3 cm (21.25 in)

Electrical

Input Voltage	104-126 Vac or 209-254 Vac
Frequency	50 Hz ± 0.5 Hz or 60 Hz ± 0.5 Hz

Environmental

Ambient Temperature	10°-43° C(50°-110° F)nominal
Relative Humidity	8% to 80% (no condensation)
Barometric Pressure	3,000 m (10,000 ft) maximum
Temperature Change Rate	6° C(10° F) per h
Disk Interchangeability Temperature Range	17° C(30° F)

PDP-11/03 SYSTEM SOFTWARE

Software systems include the operating system, programming languages, diagnostic software, paper tape software, and special-purpose software options.

Available PDP-11/03 and PDP-11/03-L system software includes:

- RT-11 Operating System, including Single Job and Foreground/Background Monitors.
- RT-11 FORTRAN
- RT-11/BASIC
- RT-11/MULTI-USER BASIC
- RT-11/FOCAL

CHAPTER 3
LSI-11/23 PROCESSOR

GENERAL

The KDF11-AA is a 16-bit, high-performance microprocessor contained on one dual-height multilayer module (M8186). The figure shows the module with its major components highlighted. Utilizing the latest MOS/LSI technology, the KDF11-AA brings the full PDP-11/34 functionality to a microprocessor that communicates along the LSI-11 bus. The KDF11-AA contains memory management as a standard feature and offers floating point as an option (KEF11-A).

The processor uses the LSI-11 bus with a new 4-level interrupt bus protocol and parity check feature. The KDF11-AA is compatible with existing LSI-11 processors and devices.

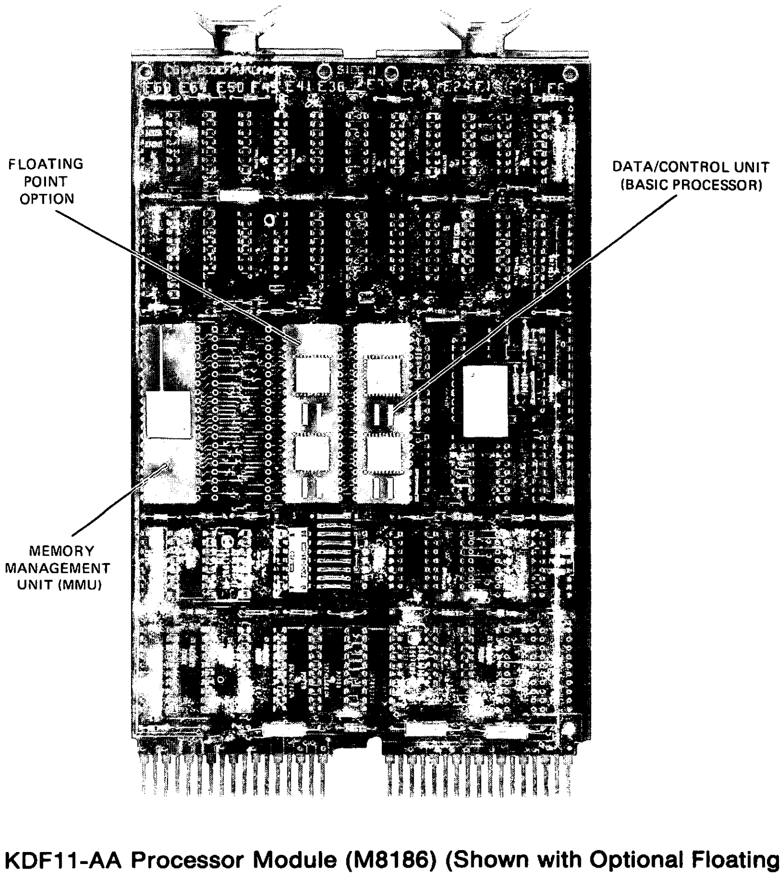
The LSI-11 bus was built around LSI technology requirements consistent with low cost, high performance, and small board form factors. Low cost and high performance are realized, in part, by using multifunction lines such as the data/address lines (DAL) that reduce the number of pins to the bus. Other lines, such as the I/O page address decode line, eliminate hardware by removing the need for identical page decoders on each interface module. A detailed description of the LSI-11 bus is contained in Chapter 8.

The KDF11-AA is software-compatible with the PDP-11 family. A wide range of software is available, including programming languages, diagnostic software, and operating systems.

FEATURES

The KDF11-AA contains the following features:

- Four-level vectored interrupts provide for fast interrupt response without device polling.
- Optional memory management for 256K bytes of protected, multiuser program space.
- Memory parity errors are recognized during every data-in bus cycle.
- Over 400 instructions for powerful and convenient programming.
- 16-bit word or 8-bit byte addressable locations.
- Eight internal general-purpose registers for use as accumulators and for operand addressing.
- Stack processing for easy handling of structured data, subroutines, and interrupts.


FLOATING POINT OPTION
DATA/CONTROL UNIT (BASIC PROCESSOR)
MEMORY MANAGEMENT UNIT (MMU)

KDF11-AA Processor Module (M8186) (Shown with Optional Floating Point)

- Asynchronous bus operation allows processor and system components (memory and peripherals) to run at their highest possible speed.
- Direct memory access (DMA) allows peripherals to access memory without interrupting processor operation.
- Modular component design allows systems to be easily configured and upgraded.
- Power fail and automatic restart hardware detect and protect against ac power fluctuations.
- Compact, double-height module size for versatile packaging.
- ODT console emulator for ease of program debugging.

SPECIFICATIONS

Identification	M8186
Size	Double
Dimensions	13.34 cm × 21.59 cm (5.25 in × 8.5 in)
Power Requirements	+5 V ± 5%, 2.0 A +12 V ± 5%, 0.2 A
Bus Loads	ac 2 unit loads dc 1 unit load
Environmental	
Storage	40° C to 65° C (104° F to 149° F) 10% to 90% relative humidity, non condensing
Operating	5° C to 60° C, (41° F to 140° F) Maximum outlet temperature rise of 5° C (9° F) above 60° C (140° F) Derate maximum temperature by 1° C (1.8° F) for each 305 m (1000 ft) above 2440 m (8000 ft).

Timing (Based on 300 ns CPU microcycle time)

(Refer to Appendix A for detailed listing of instruction times.)

Interrupt Latency (based on MSV11-D without parity, add 500 ns worst case with parity)

Worst Case	55.7 microseconds (for infrequently used instructions) 10.8 microseconds (for more frequently used group)

Typical	6.0 microseconds
Interrupt Service Time	8.2 microseconds
DMA Latency	3.49 microseconds (worst case)

DESCRIPTION

GENERAL PROCESSOR HARDWARE

The KDF11-AA processor is implemented using three chips. Two MOS/LSI chips, data and control, implement the basic processor. The memory management unit (MMU), the third chip, provides a PDP-11/34 software-compatible memory mangement scheme.

The data chip (DC302) performs all arithmetic and logical functions, handles data and address transfers with the external world, and coordinates most interchip communication. The control chip (DC303) does microprogram sequencing for PDP-11 instruction decoding and contains the control store ROM. The data and control chips are both contained in one 40-pin package. The MMU chip (DC304) contains the registers for 18-bit memory addressing and also includes the FP11 floating point registers and accumulators. Optional floating point requires the MMU chip. Data and control chips do not need the MMU chip for 16-bit addressing.

Data Chip

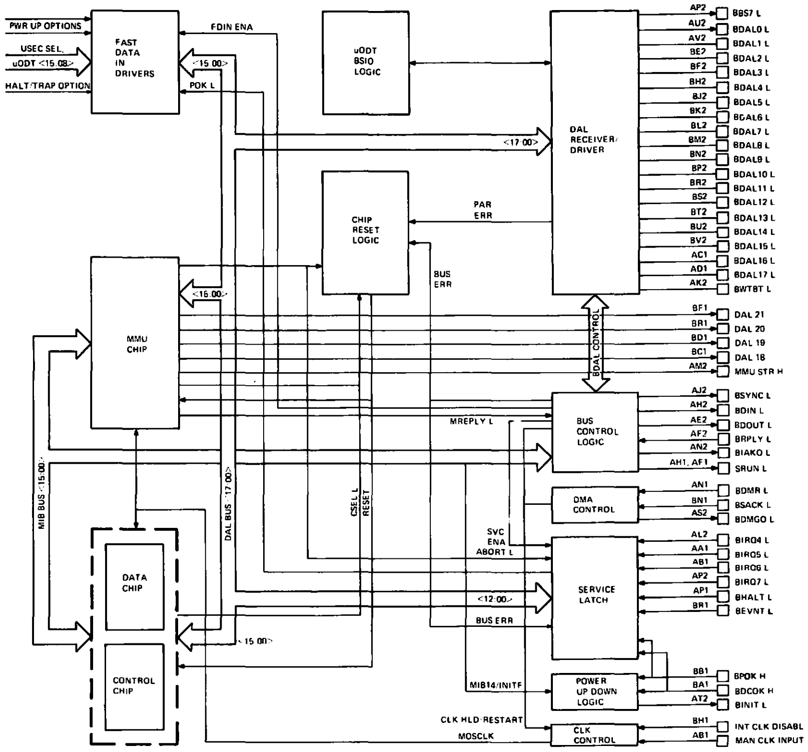
The data chip contains the PDP-11 general registers, the processor status word (PS), several working registers, the arithmetic and logic unit (ALU), and conditional branching logic. The data chip does the following.

1. Performs all arithmetic and logical functions.
2. Handles all data and address transfers with the LSI-11 bus (except relocation, which is handled by the MMU).
3. Generates most of the signals used for interchip communication and external system control.

Control Chip

The control chip contains the microprogram sequence logic and 552 words of microprogram storage in programmable logic arrays (PLA) and read-only memory (ROM) arrays.

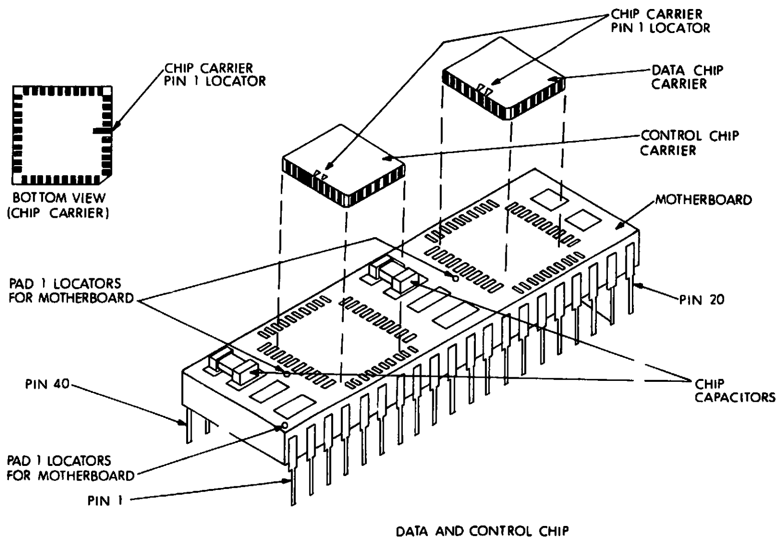
During the course of a normal microinstruction cycle, the control chip accesses the appropriate microinstruction in the PLA or ROM, sends it along the MIB to the data and MMU chips for execution, and then generates the address for the next microinstruction to be accessed. The next address is constructed from either a next address field associated with the current microinstruction or, if a microprogrammed



Processor Functional Block Diagram

branch is to be executed, the target address contained within the microinstruction itself. The control chip operation is pipelined for better performance so that the next microinstruction is being accessed while the current one is being executed. This next address is then used in conjunction with various internal status and external service inputs to determine the microprogram sequence. The control chip accesses only its local storage. However, multiple chips (up to 32) can be cascaded with external buffering to provide additional microstore.

Chip Select (CSEL) — CSEL is an open collector line which is routed to all MOS chips on the board except the MMU. The active control chip holds the line low. If a nonexistent control chip is selected by the microcode, the line is pulled high. This causes a control chip error and a trap to location 10_8>


CHIP CARRIER PIN 1 LOCATOR
DATA CHIP CARRIER
CONTROL CHIP CARRIER
MOTHERBOARD
PIN 20
CHIP CAPACITORS
CHIP CARRIER PIN 1 LOCATOR
BOTTOM VIEW (CHIP CARRIER)
PAD 1 LOCATORS FOR MOTHERBOARD
PIN 40
PAD 1 LOCATORS FOR MOTHERBOARD
PIN 1
DATA AND CONTROL CHIP

LSI-11/23 Data and Control Chip

MMU Chip

The MMU chip serves two purposes: it provides the memory management function, and it provides storage for the FP11 floating point accumulators and status registers. This chip provides dual mode (user and kernel) address relocation of 18 bits. Sixteen-bit virtual addresses are received from the data chip via the data address lines (DAL), relocated to the appropriate 18-bit physical address, and then sent on the DAL to replace the original virtual address for transmission to the external system bus. The MMU chip contains the status registers and active page registers (PAR/PDR register pairs), as well as access protection and error detection capability. The MMU chip also provides the thirty-six 16-bit registers needed for operand storage, scratchpad areas, and status information storage during floating point operations.

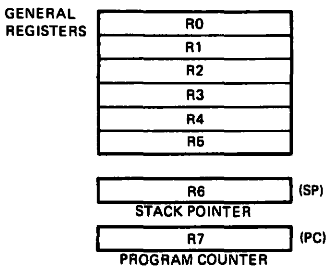
The MMU chip is controlled by information received on the microinstruction bus (MIB) from both the data chip and the control chip, and by several discrete control inputs.

The KDF11-AA can operate without the MMU chip; however, the memory would be limited to 64K bytes and the floating point registers would not be available.

General-Purpose Registers

The data chip contains eight 16-bit general-purpose registers that provide for a variety of functions. These registers can serve as accumulators, index registers, autoincrement registers, autodecrement registers, or as stack pointers for temporary storage of data. Arithmetic operations can be from one general register to another, from one memory location or device register to another, between memory locations, or between a device register and a general register. The figure identifies the eight 16-bit general registers R0 through R7.

Registers R6 and R7 are dedicated. R6 serves as the stack pointer (SP) and contains the location (address) of the last entry in the stack. Three different SP registers are used to implement the memory management feature. The highest-order PSW bits are used to select the appropriate register for the current operating mode-kernel or user. (Note: the third registers is reserved for future DIGITAL use.) Register R7 serves as the processor's program counter (PC) and contains the address of the next instruction to be executed. It is normally used for addressing purposes only and not as an accumulator. Register operations are internal to the processor and do not require bus cycles (except for instruction fetch); all memory and peripheral device data transfers do require bus cycles and longer execution time. Thus, general registers used for processor operations result in faster execution times.



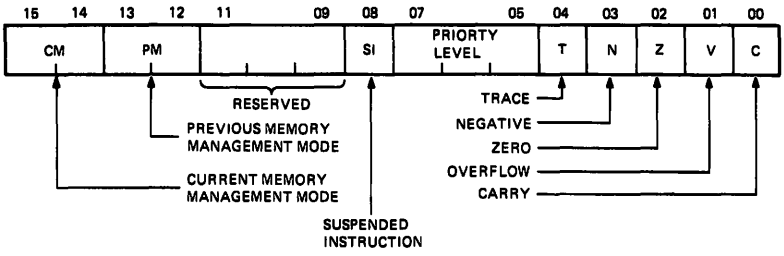
General Register

Processor Status Word (PS)

The processor status word (PS) is in the data chip and contains information on the current processor status. As the figure shows, this includes: the condition codes describing the arithmetic or logical results of the last instruction, a trace bit that forces a trap at the end of instruction execution (used during program debug), the current processor priority, an indicator of the previous memory management mode, and an indicator of the current memory management mode.

Condition Codes (PS bits 3:0) — The condition codes contain information on the result of the last CPU operation. The bits are set after execution of all arithmetic or logical single-operand or double-operand instructions. The bits are set as follows:

N = 1	if the result was negative.
Z = 1	if the result was 0.
V = 1	if the operation resulted in an arithmetic overflow.
C = 1	if the operand resulted in a carry from the MSB (most significant bit) or a 1 was shifted from MSB or LSB (least significant bit).



Processor Status Word (PS)

Trace Bit (PS bit 4) — The trace bit is used in debugging program since it allows programs to be single-instruction stepped.

Priority Level (PS bits 7:5) — These bits are used by software to determine which interrupts will be processed.

Octal Value of PS<7:5>	Interrupt Level Acknowledged*
7	none
6	7,
5	7, 6,
4	7, 6, 5,
3	7, 6, 5, 4
2	7, 6, 5, 4
1	7, 6, 5, 4
0	7, 6, 5, 4

* Higher levels acknowledged first.

Suspended Instruction (SI) (PS bit 8) — This bit is reserved for DIGITAL use and is intended for future optional instruction sets. This bit is read/write and has no protection mechanism.

Previous Mode (PS bits 13:12) — These bits are used with memory mangement to indicate what the last memory management mode was. They are read/write bits and are present even without the memory management option.

Current Mode (PS bits 15:14) — These bits indicate what the present memory management mode is. They are read/write and are present even without the memory management option.

Memory Management

Memory management has the following three major features:

1. Two software modes that are useful for multiuser (timesharing) systems.
2. Extended physical addressing (greater than 64K bytes, up to 256K bytes) for allowing more than one program to reside in memory at the same time.
3. Memory protection for controlling user program access to system resources (e.g., memory, I/O).

The first feature has two software modes, kernel and user. Kernel mode is employed for executing a user program and restricts processor privileges (e.g., HALT instruction cannot be executed). The second feature utilizes mapping registers to map the 64K-bytes virtual address space anywhere in the 256K-bytes physical address space. The third feature allows restricted access to virtual memory pages (a page is between 0 and 8K bytes long). This permits the operating system software rather than user programs to control system resources. Chapter 9 contains a complete discussion of memory management.

INSTRUCTION SET

The KDF11-AA instruction set provides over 400 powerful instructions. As a comparison, consider that most other (i.e., accumulator-oriented) 16-bit processors require three separate instructions to execute a common double-operand instruction (e.g., ADD).

Conventional Approach

LDA A	Load contents of memory location A into accumulator.
ADD B	Add contents of memory location B to accumulator.
STA B	Store result at location B.

By constrast, the KDF11-AA can fetch both operands, execute, and store the result in one instruction.

KDF11-AA Approach

ADD A, B	Add contents of location A to location B; store results at location B.

This greater efficiency not only saves memory space and time, but also improves processor speed since fewer instruction fetches are required.

Another major advantage to the KDF11-AA instruction set is the absence of special-purpose input/output instructions. Special I/O instructions are unnecessary since peripheral device registers are accessed in the same way as main memory locations. This approach to handling I/O devices allows the normal instruction set to be used to test and/or manipulate the various I/O device register bits. For example, a compare instruction can test status bits directly in the I/O device register without bringing them into memory or disturbing any of the general registers; control bits can be set, cleared, or shifted as is most convenient; and peripheral data can be arithmetically or logically altered when received at the device register and before being stored in memory. Refer to Chapter 7 for a complete description of the instruction set and its utilization.

Addressing Modes

Much of the flexibility of the KDF11-AA is derived from its wide range of addressing capabilities. Addressing modes include sequential forward or backward addressing, address indexing, indirect addressing, absolute 16-bit word and 8-bit byte addressing, and stack addressing. Variable-length instruction formatting allows a minimum number of words to be used for each addressing mode. The result is efficient use of program storage space. For more details on addressing modes, refer to Chapter 6.

FLOATING POINT OPTION

Forty-six floating point instructions are available as a microcode option (KEF11-A) on the KDF11-AA processor. These instructions supplement the integer arithmetic instructions (e.g., MUL, DIV, etc.) in the basic instruction set. The floating point option allows floating point operations to be executed 5 to 10 times faster than equivalent software routines and provides for both single precision (32-bit) and double precision (64-bit) operands. This option also conserves memory space, since floating point routines are executed in microcode instead of software. This option implements the same floating point instruction set found on the PDP-11/34, -11/60, and -11/70. For a complete description refer to Chapter 10.

DATA-ADDRESS LINES (DAL)
The DAL bus is routed between all the MOS chips, along the processor board, and to the LSI-11 bus transceivers. The 16-bit DAL bus is time-multiplexed. During clock-high time, the DAL bus transfers data from the data chip to the other MOS chips or between the processor board and the MOS chips. During clock-low time, the DAL bus transfers service data (external and internal interrupt requests) from the board to the control chip. (The control chip receives service information and determines whether to interrupt or fetch the next instruction.)

MICROINSTRUCTION BUS (MIB)
The 16-bit microinstruction bus is common to all data and control chips. A subset of the MIB is routed to the MMU because it does not need access to all MIB control signals. A different subset of the MIB controls the processor board logic.

The MIB is time-multiplexed and is used for different functions during clock high and low times. During clock-high time, the MIB transfers control information from the data chip to all control chips, the MMU, and the board logic. During clock-low time, the MIB transfers microinstructions from the active control chip to other control chips and the data chip.

MIB15/Memory Management Enable (MME)
During clock-high time, MIB15 carries MME from the MMU chip. MME is an active low signal. After being pulled low by the MMU chip, MME indicates to the processor board logic that a relocated-address microcycle should be performed. MME is also asserted low by the processor board during console ODT to allow access to greater than 32K words of memory without using the MMU chip.

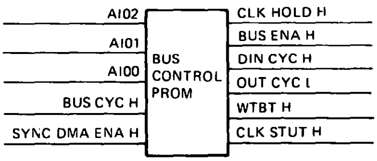
MIB14/Initialize (INIT F)
During clock-high time, MIB14 contains an active low initialize signal (INIT F) used by the board logic to generate BINIT L. At the end of every clock-high time, the processor monitors INIT F. If INIT F is asserted low, the processor generates BINIT L onto the LSI-11 bus. DINIT L holds the INIT F flip-flop in the 0 state during power-up so that BINIT L is constantly driven onto the LSI-11 bus until DCOK H from the power supply goes high.

MIB13/Interrupt Acknowledge (IAK)
MIB13 contains IAK during clock-high time, and is used to generate BIAK L onto the LSI-11 bus. The highest priority device that is

requesting an interrupt uses BIAK L and BDIN L as a signal to assert its interrupt vector on the LSI-11 bus. IAK occurs only during an input vector microcycle.

MIB12, 9, 8/Address-Input-Output (AIO) Codes
These three control lines along with two other signals, BUS CYC H and SYNC/DMA ENA H, are fed into the bus control PROM as shown in the Bus Control PROM figure. The PROM decodes them to determine the type of microcycle currently executing within the MOS chips. The PROM output's various control signals which perform the following functions.

1. CLK HOLD H stops the clock generator in the high state for asynchronous data transfers. This signal stops the clock while waiting for BRPLY and during address cycles if a previous bus cycle is not complete or if some other device is Bus Master.
2. BUS ENA H enables LSI-11 bus drivers during address and data-out bus cycles only.
3. DIN CYC H drives the BDIN L bus driver.
4. OUT CYC H drives the BDOUT L bus driver.
5. WTBT H drives BWTBT L bus signal whenever an address microcycle is followed by a data-out microcycle and whenever a byte data transfer is in progress.
6. CLK STUT H, for clock control, is used to extend the clock-high time of address microcycles and nonbus data-in and data-out microcycles.



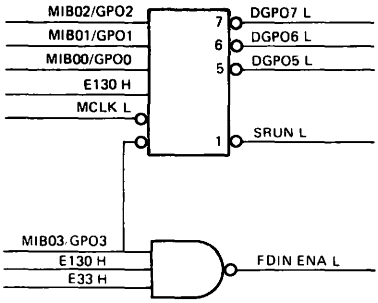
Bus Control PROM

BUS CYC H — This signal is a function of the sync signal from the data chip (SYNCF). If a data transfer to or from the data chip is internal to the MOS chip set, then BUS CYC H is low. If it is an external bus transfer, then BUS CYC H is high. In the case of internal data transfers, the clock is lengthened one clock tick to allow the chip set more time to complete its internal transfer. In the case of bus-type data transfers, the bus drivers (DOUT transfers) or receivers (DIN transfers) are enabled, and the master clock is halted in the high state, waiting for BREPLY L from the bus.

SYNC/DMA ENA H — SYNC/DMA ENA H indicates that another peripheral is still bus master or that the last bus cycle is not yet complete. Its function is to prevent the MOS chip set from attempting to use the LSI-11 bus when the bus is still being used.

The master clock is halted during LSI-11 bus data transfers while transferring data to the peripheral or receiving data from the peripheral. Once this is accomplished, the master clock starts up again and microinstructions are again executed. Concurrently, the processor is terminating the previous bus cycle. Because the processor cannot terminate the cycle until BREPLY has been deasserted by the peripheral (there is no time limit on this action taking place according to LSI-11 bus protocol), it is possible for the previous bus cycle to still be active when the chip set is ready for the next bus cycle. SYNC/DMA ENA H causes the clock to stop in the address cycle in this case and halts the chip set in the address microcycle until the previous bus cycle is properly completed (BSYNC L negated).

MIB03/GPO 3 — Control code GPO 3, driven by the data chip, is detected by the GPO decode logic and properly timed to produce FDIN ENA L. This signal is used to gate power-up information from the jumpers on the processor board.



GPO Decode Logic

MIB02, 01, 00/GPO 2, 1, 0 — GPO 2, 1 and 0 are driven by the data chip during clock-high time and perform control functions on the processor board. These signals are decoded by the logic shown in the GPO Decode Logic figure. The decoded output is shown in the General-Purpose Output Signals table.

BSYNC L Logic

The logic shown in the BUS SYNC Logic figure controls the assertion of BSYNC L onto the LSI-11 bus. The start of all bus cycles <(DATI, DATO(B), DATIO(B))> is signaled by SYNCF L going low on MIB07 of the data chip during clock-high time. SYNCF L is clocked into both the BUS CYC flip-flop, and the SYNCF flip-flop at the end of the clock-high time. A set BUS CYC flip-flop indicates to the DMA logic that the processor is going to use the bus, and therefore a DMA request cannot be granted.

General-Purpose Output Signals

GP02	GP01	GP00	Output Name	Function
1	1	1	DGP07 L	Loads the two highest order address bits into a latch while in micro-ODT. This allows 18-bit addressing to be accomplished without using the memory management unit while in ODT.
1	1	0	DPG06 L	Clears the power-fail flip-flop after the power-fail sequence has been executed in microcode.
1	0	1	DPG05 L	Clears the event flip-flop after the even interrupt has been serviced in microcode.
0	0	1	SRUN L	Generates a low-going pulse that is routed directly to edge fingers AF1, AH1 whenever a character is received from the serial line unit while in micro-ODT. This signal can be used to cause a steady RUN

indication while the processor is executing microinstructions and a flashing indication when typing characters in console-ODT.

The SYNC flip-flop feeds the BSYNC flip-flop. This flip-flop is strobed every microcycle, 33 ns after the start of clock-high time. Thus, the BSYNC flip-flop will be set 33 ns into clock-high time of the microcycle after the address microcycle. This delay is necessary to allow sufficient address set-up time on the bus. Once the BSYNC flip-flop is set, it drives the bus transceiver and asserts BSYNC L onto the LSI-11 bus.

Once the BSYNC flip-flop is set, it remains set until the LSI bus completes the bus cycle. The SYNCF signal from the data chip clears on a data-in or data-out microcycle. The BSYNC Reset logic uses SYNC REP L and RESTARTEND, both functions of BRPLYL, to clear BSYNC L after the rising edge of BRPLYL. BUS CYC L and DOUT BLOCK L block the BSYNC flip-flop from being cleared after the DATI portion of a DATIO cycle.

These signals also prevent the BSYNC flip-flop from being cleared for at least 175 ns after BDOUT L is cleared (as per bus specifications). SYNC RESET L clears the BSYNC flip-flop on power-up if a bus time-out occurs, and prevents it from setting when an MMU abort occurs.

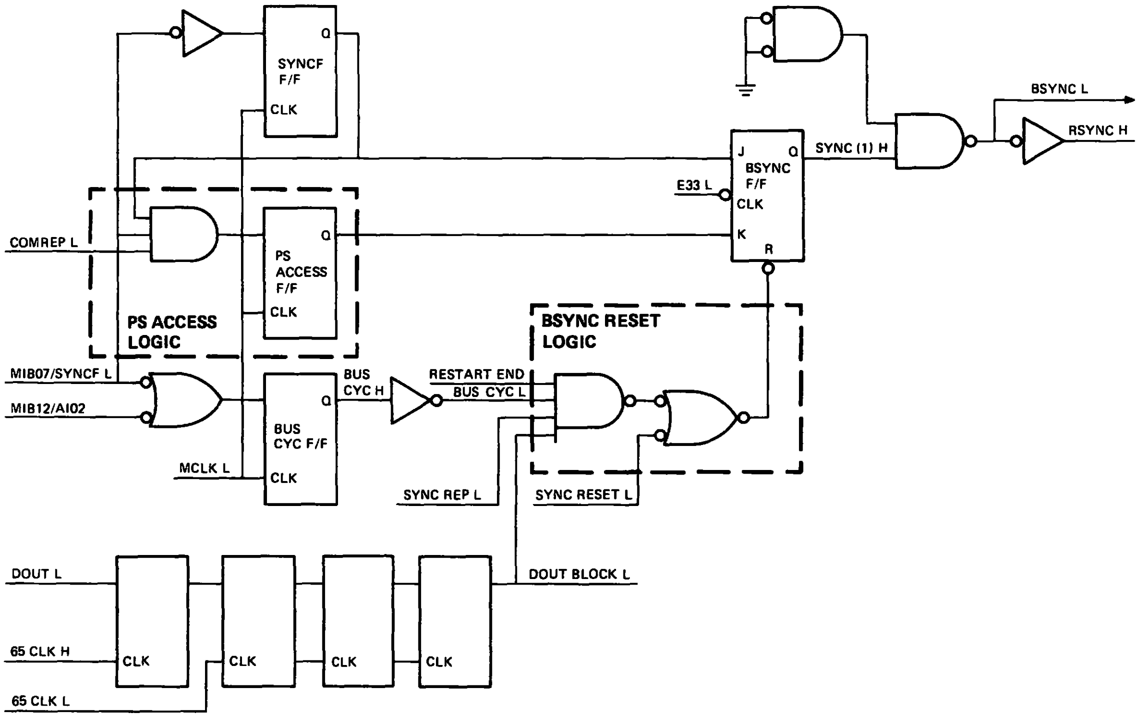
PS Access Logic

The PS (processor status word) access logic feeds the K input of the BSYNC flip-flop and is used only when the PS is accessed. The PS is contained in the data chip. When 777776_8(the address of the PS in the data chip) appears on the DAL during an address microcycle, the data chip decodes the address and access to the PS is allowed. The bus cycle is terminated by deasserting the SYNCF line without allowing a DATI or DATO AIO code.

The PS access flip-flop stores this condition until the start of the next clock-high time. This signal is fed to the K input of the BSYNC flip-flop and resets BSYNC at the start of the next microcycle.

DIRECT MEMORY ACCESS (DMA)

DMA on the KDF11-AA board allows peripherals to gain control of the LSI-11 bus from the processor and transfer data directly between a peripheral and memory. In this way, data transfers can occur at the full memory speed rather than having the processor transfer data words one at a time between the peripheral and memory. A speed gain of



BUS SYNC Logic

about 12 to 1 over regular programmed transfers is gained by this technique.

The signals required for the DMA logic are the following.

BDMR L	This is the DMA request signal. A peripheral device asserts this line when it is ready to use the bus for a DMA transfer. This line is common to all peripheral devices.
BDMGO L	This DMA grant signal is issued by the processor in response to a DMA request. By asserting this line, the processor indicates that it will halt processing as soon as the current bus cycle is completed. The processor will also disable all bus control lines and data-address lines (BDAL) so that the peripheral device can use them to control the bus. The BDMR line is common to all peripheral devices. BDMGO L is a daisy-chained signal. Any memory or peripheral device that does not want to use the bus simply passes the signal on. The first (physically closest to the processor) device on the bus desiring to use the bus "takes the grant;" i.e., blocks the signal from being passed on. Therefore the peripheral closest to the processor requesting the bus at the time the grant is issued gets to use the bus. In order to prevent hogging of the bus by peripheral devices nearest the processor, DMA transfer time must be as short as possible.
BSACK L	This DMA acknowledge signal is issued by the peripheral device taking control of the bus. This

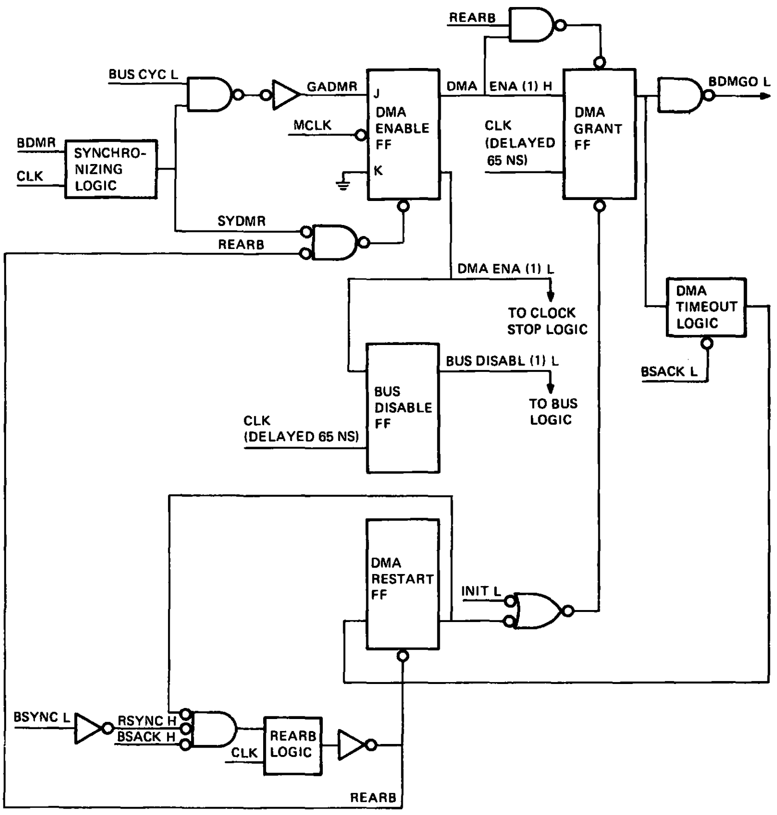
	signal completes the handshake between the processor and the peripheral device and indicates to the processor that a device has taken the bus.
No SACK Timeout	In LSI-11 bus systems there is a possibility that a device can request use of the bus and then not take the DMA grant signal. The no SACK timeout feature clears the DMA grant signal and returns bus mastership to the processor if no peripheral device has issued BSACK L within 18 microseconds after the processor has issued a grant. This prevents a potential bus lockup problem in which the processor has given up the bus but no one has taken the grant.

DMA Logic
The DMA logic is shown in the figure. BDMR L signals are received from the bus on edge pin AN 1 and synchronized with the processor high-frequency clock through a high-speed synchronizer. This signal is called SYDMR for "synchronized DMR." The SYDMR signal is gated with the signal BUS CYC L to block DMA requests from reaching the DMA ENA flip-flop when a bus cycle is in progress. The gated signal is called GADMR for "gated DMR." The DMA ENA flip-flop samples the GADMR line at the beginning of every clock-high time (about every 290 ns). When a valid GADMR is latched into the DMA ENA flip-flop, the DMA cycle is started. Note that DMA request is always taken unless the processor is currently in a bus cycle. This is necessary to provide fast response to DMA requests.

Once DMA ENA is latched, DMA grant is issued on the LSI-11 bus approximately 65 ns later by the DMA ENA H being clocked into the DMA grant flip-flop. Granting the DMA request also starts the timer which is set for 18 microseconds. At exactly the same time DMA grant is enabled, the DMA bus disable flip-flop disables the BDAL bus drivers on the processor board. The DMA ENA (1) L signal also blocks any further clock restarts from occurring until the DMA cycle that is just starting is completed. It does this by blocking the AND inputs to the clock restart logic.

Once DMA grant is issued, the processor board waits for a BSACK L signal indicating that a peripheral device has taken the DMA grant. The BSACK L line is monitored by a bus receiver; an active BSACK L resets the no SACK timeout timer which clocks a 1 into the DMA restart flip-flop. The DMA restart flip-flop is now armed.

As soon as the bus is given up by the current DMA master, this flip-flop will allow the DMA rearbitration process to restart. This occurs when BSACK L and BSYNC L are deasserted as the bus master gives up the bus. These signals, along with the armed DMA restart flip-flop, satisfy the gate which feeds the rearbitration logic and a re-start/rearbitration takes place.



DMA Logic

According to system protocol, the processor is the lowest priority bus master. When a bus master give up the bus, the processor should immediately check for another pending request. If another request is pending, another BDMGO is reissued and a new peripheral takes control. In the KDF11-AA, rearbitration takes place each time the bus is given up. If DMA requests are arriving at too great a rate, it is possible to have the processor constantly arbitrating among bus masters. This effect can be illustrated by holding the BDMR L line low which blocks any instruction fetches by the processor.

DMA Latency

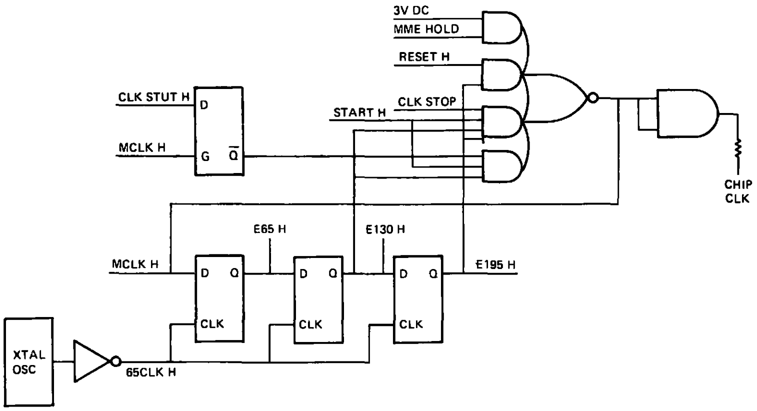
DMA latency is the time from when the DMA request arrives at the processor until BDMGO is put on the bus. The maximum DMA latency is important because of data loss problems. For exampe, once the heads of a disk drive are over the proper sector, the disk controller must become bus master within a certain period of time. If it does not, information will overflow the temporary data buffers in the disk drive interface and cause data-late errors. Since the KDF11-AA does not grant bus mastership during ongoing bus cycles, worst-case DMA latency occurs when the DMA request arrives just before the start of the longest bus cycle (DATIO). In this case the grant will be issued after the cycle has completed.

CLOCK GENERATOR CIRCUITRY

The KDF11 chip set clock can be suspended in the high state indefinitely, but can only remain in the clock low state for a limited period of time to avoid loss of internal chip data. A twisted ring oscillator, shown in the figure, is used with a high-frequency crystal clock input to generate the required clcok signals that control the MOS/LSI chips. The TTL level output of the ring oscillator (MCLK H) is driven through a high-voltage clock buffer/driver to produce the high-voltage CHIP CLK that drives the MOS chips.

Initialization

When the processor receives +5 Vdc and +12 Vdc, the ring oscillator is initialized and held in this state until BDCOK H is asserted by the power supply (or the wake-up circuit). The initialization circuity is shown in the figure. The output of the second stage of the DCOK H synchronizer circuit holds START H low. The processor board initializes with MCLK H = 1 and all three stages of the ring oscillator also equal 1 (E65H, E130H, E195H). When DCOK H goes high, it is first synchronized with the high-frequency clock (65CLK H) and then releases the ring oscillator from its initialized state. The synchronizer is necessary because DCOK H is asynchronous to any circuitry on the processor board and feeding DCOK H directly into the ring oscillator could lead to a truncated first cycle of the processor. Once the oscillator is freed, it immediately causes MCLK H to go low and enters the clock-low state.



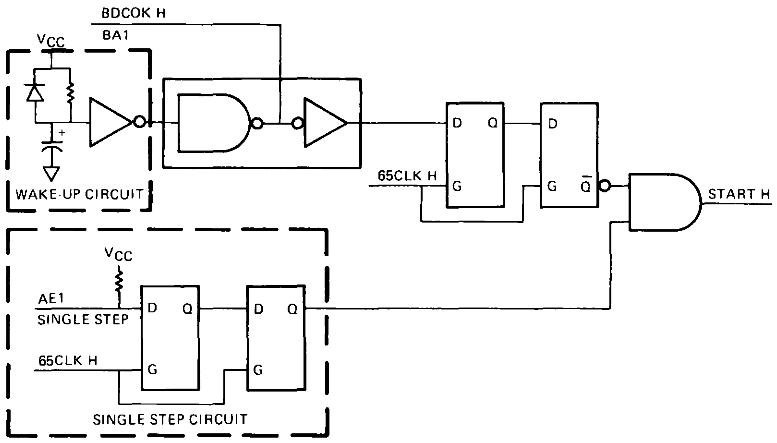
Clock Generator

Wake-Up Circuit

The "wake-up" circuit on the KDF11-AA module consists of a diode, a resistor, a capacitor, and a Schmidt trigger inverter, all shown at the left in the figure. This circuit provides automatic generation of BDCOK H 50 ms after the +5 V supply is turned on. For the circuit to function, the +12 V must be applied before or at the same time the +5 V is applied, and the rise time of the +5 V supply must be on greater than 50 ms.

Single-Step Circuit

The single-step circuit is shown in the lower portion of the figure. This circuit can be used in conjunction with an external circuit to stop the processor (i.e., hold the clock high indefinitely) in the bus data-in or data-out part of the cycle at a selected address. The external circuit must monitor the BDAL line and compare the address issued by the processor at BSYNC L time with a desired stop address or addresses. If a valid compare occurs, the external circuit should pull SINGLE STEP to a logic low level as soon as BDIN L or BDOUT L appears on the bus. The processor will then stop in the bus data-in or data-out microcycle and the data driven from the processor (in the case of data-out, data-in transfers) can be observed on the BDAL lines and any other internal points of the system can be probed manually. The processor can be released from this state by releasing the single-step line and will resume executing instructions.



Clock Generator Initialization Circuitry

CLOCK GENERATOR CYCLES

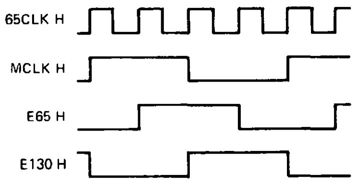
The clock generator is capable of producing a normal cycle and four variations of the normal cycle used for special functions.

Normal Cycle

The normal cycle consists of two cycles of the high-frequency clock in the high state and two cycles in the low state. For this type of cycle, START H is constantly high, RESET H is low, and CLK STOP is low. The figure shows this cycle.

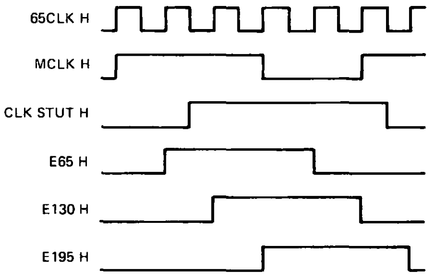
Clock Stutter Cycle

The clock stutter cycle is generated on all address microcycles and for all internal data transfers among the MOS chips. It is the same as the normal cycle discussed above except that the clock-high time is extended from two cycles of the high-frequency clock to three. This stretched or "stuttered" clock time allows the DAL lines to settle before the address is driven out onto the bus. The cycle also allows extra time for data transfers between MOS chips.



Normal Clock Cycle

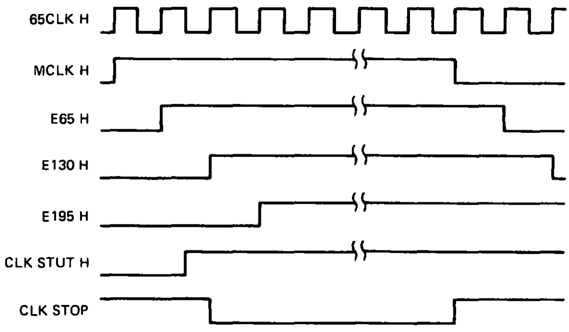
The cycle is generated by the CLK STUT H signal from the bus control PROM being fed through a transparent latch that is enabled during phase time. The output of the latch inhibits the E130 H input to the feedback loop from causing MCLK H to go low. Instead, the ring oscillator output drops when E195 H goes high, one cycle of the high-frequency clock later. The stutter cycle is shown in the figure.



Clock Stutter Cycle

Clock Stop Cycle

The clock stop cycle is generated during bus data-in and bus data-out transfers when the chip set must wait for a REPLY from the LSI-11 bus before it can continue. It is also used to prevent the chip set from continuing past the address microcycle portion of a bus cycle when a DMA device has bus "mastership." For a clock stop cycle, the bus control PROM generates CLK STUT H and CLK HOLD H. The CLK STUT H signal stretches the clock-high time from two to three high-frequency clock cycles. The CLK HOLD H signal is clocked into a flip-flop (the CLK STOP flip-flop) every cycle after two cycles of the high-frequency clock. The output of this flip-flop, CLK STOP, goes low and holds MCLK H in the high state until the CLK STOP flip-flop is cleared. In the case of a bus data-in or data-out cycle, the flip-flop is cleared 200 ns after REPLY has been received from the addressed device, or, in the DMA case, 130 ns after the DMA device has given bus masterhip back to the processor. This cycle is shown in the figure.



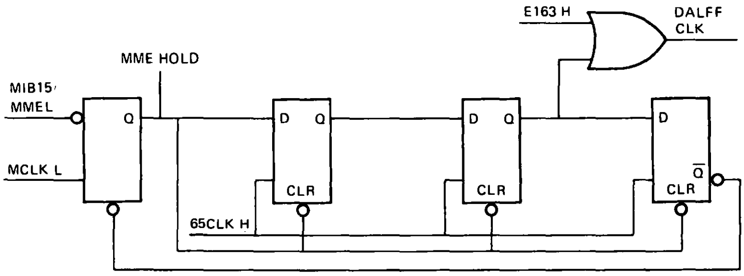
Clock Stop Cycle

Memory Management Cycle

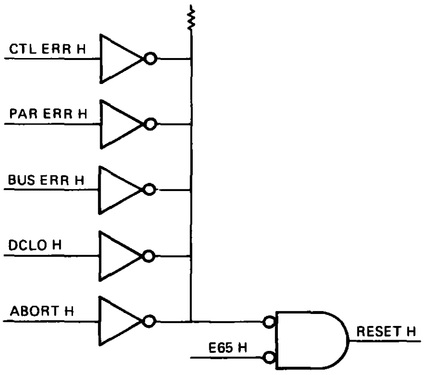
This cycle occurs during address microcycles when the memory management chip is present and is enabled to do address relocation (enabling of the MMU is under software control). The MMU chip signals to the processor board that it wants to do address relocation by asserting the MIB line MME L at the end of clock-high time of an address microcycle. The relocation circuit, shown in the figure, detects the MME L signal and causes MME HOLD to be asserted high 65 ns into clock-low time of the address microcycle. MME HOLD holds MCLK in the clock low state for a total of five high-frequency clock periods or 325 ns. A pulse is produced 195 ns into clock-low time which passes through the OR gate and causes DALFF CLK to latch the relocated address, driven out of the MMU chip onto the DAL bus at this time, into the DAL driver flip-flops. Since the BDAL bus is continuously enabled during this time, the relocated address is immediately driven onto the BDAL lines. The relocation timing circuitry automatically clears itself after five high-frequency clock periods and releases MME HOLD which immediately allows MCLK H to go high, ending clock-low time.

Reset Cycle

The final variation of the basic cycle is when a CHIP RESET occurs. CHIP RESET is generated by the circuit shown in the figure and occurs for any one of five error conditions that warrant immediate attention by the chip set. RESET H is enabled 65 ns into clock-low time and causes the ring oscillator to stretch clock-low time from two periods of the high-frequency clock to three. This extended clock-low time allows CHIP RESET to initialize the MOS/LSI chips.



Relocation Timing Circuit



Reset Circuit

Chip Reset/RESET

RESET is routed to all MOS chips except the MMU. If an interrupt requiring immediate attention occurs, the line is asserted high. The following five interrupts require immediate attention.

1. Control error — Nonexistent control chip selected by the micro-code. A trap to location 10_8 occurs.

2. Bus error — Nonexistent memory location accessed. A trap to location 4_8occurs.

3. Parity error — A parity error detected on a current read from memory. A trap to location 114_8occurs.

4. MMU abort — The MMU has aborted a mapped reference. A trap to location 250_8occurs for any of the following reasons.
   - The memory location referenced is not present in the current user's protected address space.
   - An attempt is made to modify a write-protected location.
   - The user is exceeding his allotted page boundary.

5. DC Power-Up — Upon power-up the processor forces two sequential RESETS to the chip set to initialize all internal chip registers. The dc power-up line than clears and is not activated again while dc power is on.

CONFIGURATION

JUMPER CONFIGURATIONS

Several jumpers on the processor module provide user-selectable features. The following table lists the jumper configurations and the accompanying figure shows the location of these jumpers. Jumpers not discussed are reserved for use by DIGITAL and should not be used.

Jumper Configurations

Jumper	Name	In	Out
W1	Master clock	Enable internal master clock	Do not remove. Manufacturing use only
W2	Reserved for DIGITAL use	Factory-installed	Do not remove
W4	Event line enable	Disabled	Enabled
W5,W6	Power-up mode selector	See text	See text
W7	Halt/trap option	Trap to 10_8 on halt	Enter console ODT on halt
W8	Conventional bootstrap start address, enable if power-up mode 2 is selected	Power-up to bootstrap address 173000_8	Power-up to bootstrap address selected by jumpers W9-W15
W9-W15	User-selectable bootstrap starting address for power-up mode 2	See text	See text
W16	Reserved for DIGITAL use	Must be installed	Do not remove
W17	Reserved for DIGITAL use	Must be installed	Do not remove
W18	Reserved for DIGITAL use	Must be installed	Do not remove

Master Clock — W1
The internal 13.8 MHz oscillator is disconnected from the clock circuitry if W1 is removed. This jumper is used by DIGITAL manufacturing and is not to be removed by the user.

Power-Up Mode Selection — W5 and W6

Four power-up modes are available for user selection. Selection is made by removal or insertion of jumpers W5 and W6 as shown in the following listing.

Mode	Name	W6*	W5*
0	PC@24, PS@26	R	R
1	Console ODT	R	I
2	Bootstrap	I	R
3	Extended microcode	I	I

*R = jumper removed; I = jumper installed.

Only the power-up mode is affected, not the power-down sequence. The following paragraphs describe the sequence of events after executing common power-up, when selecting each of the four modes. The state of bus signal BHALT L is significant in power-up mode operation.

Power-Up Mode 0 (PC@24, PS@26)

This mode causes the microcode to fetch the contents of memory locations 24_8 and 26_8 and loads their contents into the PC and PS, respectively. The microcode then examines BHALT L. If BHALT L is asserted, the processor enters console ODT mode. If BHALT L is not asserted, the processor begins program execution by fetching an instruction from the location pointed to by the PC. This mode is useful when power fail/auto restart capability is desired.

Power-Up Mode 1 (Console ODT)

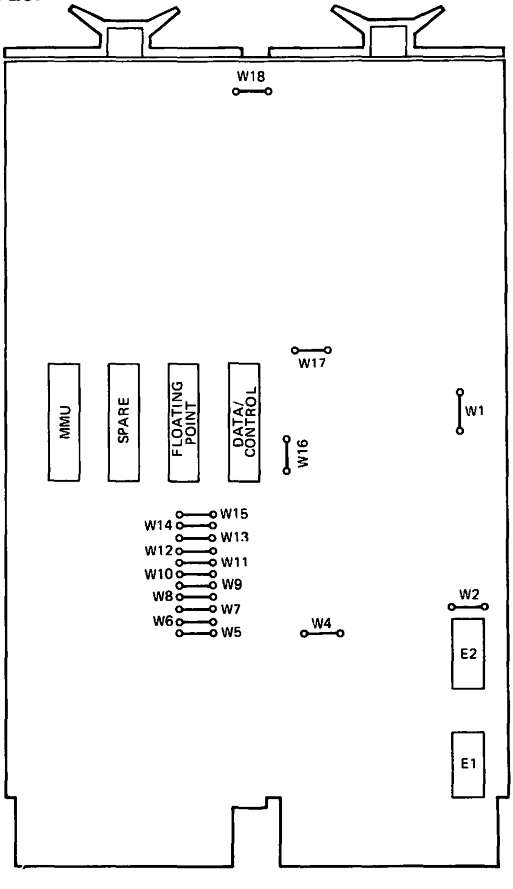
This mode causes the processor to enter console ODT mode immediately after power-up regardless of the state of any service signals. This mode is useful in a program development or hardware debug environment, giving the user immediate control over the system after power-up.

Power-Up Mode 2 (User Bootstrap Starting Address Shown by W8-W15)

This mode causes the processor to internally generate a bootstrap starting address by looking at jumpers W8 through W15. This address is loaded into the PC. The processor sets the PS to 340_8 (PS <07:05> = 7_8) to inhibit interrupts before the processor is ready for them. If BHALT L is asserted, the processor enters console ODT mode. If not, the processor begins execution by fetching an instruction from the location pointed to by the PC. This mode is useful for turnkey applications where the system automatically begins operation without operator intervention.

Event Line — W4
The bus signal BEVENT L causes the event line flip-flop to be set. When the processor enters the service state the request will be honored if the PS <07:05> is 5 or less. (BEVENT is a level 6 interrupt.) This causes the microcode to clear the request flip-flop and trap to the line clock vector (location 100_8). If W4 is inserted, the request flip-flop is disabled and therefore the BEVENT signal is disabled. Users would disable BEVENT, which is normally used as a 60 Hz real-time clock, if they have a programmable clock on the LSI-11 bus.

NOTE
The LSI-11 and LSI-11/2 processors treat a BEVENT interrupt at a different priority level than the LSI-11/23.



KDF11-AA Jumper Locations

Power-Up Mode 3 (Microcode — For Future Use)
This mode causes the microcode to jump to optional control chip 37_8, location 76_8, and begin microcode execution. This mode is reserved for future DIGITAL use and is not recommended for customer usage. If it is erroneously selected, the processor will treat it as a reserved instruction trap to location 10_8.

Halt/Trap Option — W7
If the processor is in kernel mode and decodes a HALT instruction, BPOK H is tested. If BPOK H is negated, the processor will continue to test for BPOK H. The processor will perform a normal power-up sequence if BPOK H becomes asserted sometime later. If BPOK H is asserted after the HALT instruction decode, the halt/trap jumper (W7) is tested. If the jumper is removed, the processor enters console ODT mode. If the jumper is installed, a trap to location 10_8 will occur.

NOTE

In user mode a HALT instruction execution will always result in a trap to location 10_8.

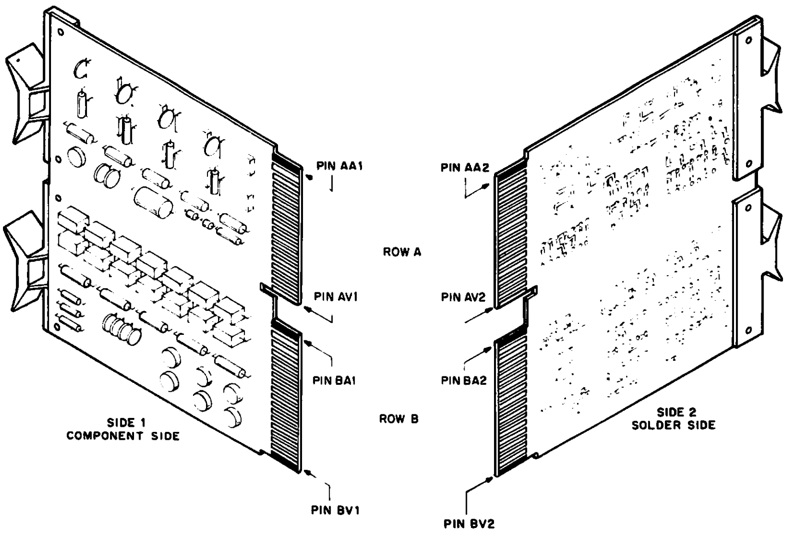
This feature is intended for situations, such as unattended operation, where recovery from erroneous HALT instructions is desirable.

Starting Address 173000_8 —W8
When power-up mode 2 is selected, the processor examines jumper W8 to determine the starting address for program execution. If W8 and a compatible bootstrap module such as BDV-11 are installed in the system, the microcode will begin execution at 173000_8 (conventional starting address for DIGITAL systems). If W8 is removed, a trap to 4_8 (nonexistent address) will occur. If W8 is removed, the processor looks at jumpers W9 through W15 for the starting address.

Selectable Starting Address — W9 through W15
If the user wishes to start execution from an address other than 173000_8, jumpers W9 through W15 can be used to specify the high byte <15:09> of the starting address. Jumpers W15 through W9 correspond to address bits <15:09>, respectively. Bits <08:00> of the starting address are set to 0 by the processor. Jumpers are installed for logic 1, removed for logic 0. The starting address can reside on any 256-word boundary in the lower 32K of memory address space.

MODULE CONTACT FINGER IDENTIFICATION
DIGITAL plug-in modules, including the KDF11-AA, 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 the figure below, 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. A typical pin is designated as follows.



Double-Height Module Contact Finger Identification

BE2

Slot (Row) Identifier
"Slot B"

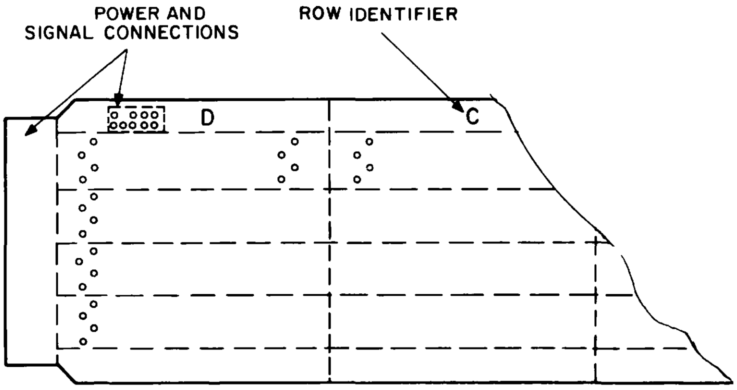
Module Side
Identifier
"Side 2" (solder side)

Pin Identifier
"Pin E"

The positioning notch between the two rows of pins mates with a protrusion on the connector block for correct module positioning.

BACKPLANE PIN ASSIGNMENTS AND LSI-11/23 UTILIZATION
When configuring a system with the LSI-11/23, the module may be inserted in one of several available backplanes. Using a typical backplane as an example, the accompanying figure shows the backplane pin identification. Individual connector pins shown are viewed from the underside (wiring side). Only pins for one bus location (two slots) are

shown in detail. This pin pattern is repeated eight times on this backplane, allowing the user to install several double-height modules.



Typical Backplane Pin Identification (Pin Side View Shown)

HARDWARE OPTIONS

PDP-11/23 systems can be configured using a variety of backplanes, power supp'ies, enclosures, memories, peripherals, etc.

Backplanes

Any of the following LSI-11 bus-compatible backplanes can be used with the LSI-11/23.

- H9270 — Accepts quad- or double-height modules
- H9273-A — Accepts quad- or double-height modules
- H9281 — Accepts double-height modules only
- DDV11-B — Accepts quad- or double-height modules

H9270 Backplane — The H9270 consists of an 8-slot backplane with a card guide assembly. This backplane is designed to accept up to eight double-height modules (including processor), four quad modules, or a combination of quad- and double-height modules. When used for bus expansion in multiple backplane systems, the H9270 provides space for up to six option modules, plus the required expansion cable connector module(s) and/or terminator module.

VIEW FROM MODULE SIDE OF BACKPLANE

PROCESSOR (HIGHEST PRIORITY LOCATION)	PROCESSOR OR OPTION 1	1
OPTION 3	OPTION 2	2
OPTION 4	OPTION 5	3
OPTION 7 (LOWEST PRIORITY LOCATION)	OPTION 6	4

H9270 Options Positions

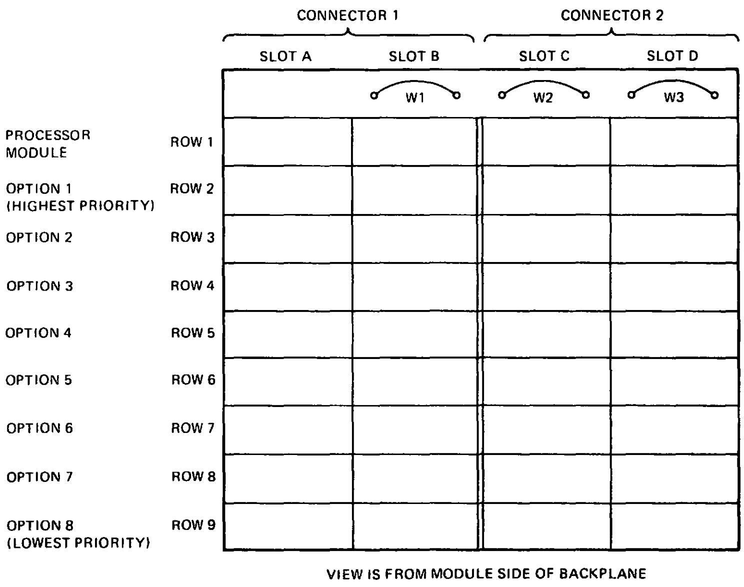
H9273-A Backplane — The H9273-A backplane logic assembly consists of a 9 × 4 backplane (nine rows of four slots each) and a card frame assembly. Power and signals are supplied to the backplane through connectors J7 and J8.

The H9273-A backplane is designed to accept both double-height and quad-height modules with the exception of the MMV11-A core memory module. The backplane structure is unique in that it provides two distinct buses: the LSI-11 bus signals (slots A and B) and the CD bus (slots C and D). The connectors that comprise this backplane are arranged in nine rows. Each connector has two slots, each of which contains 36 pins, 18 on either side of the slot.

Three jumpers (W1, W2, and W3) are shown in the following figure. Jumper W1 enables the line-time clock when inserted and disables it when removed.

NOTE

Only one BA11-N mounting box in any system may have the line-time clock enabled.

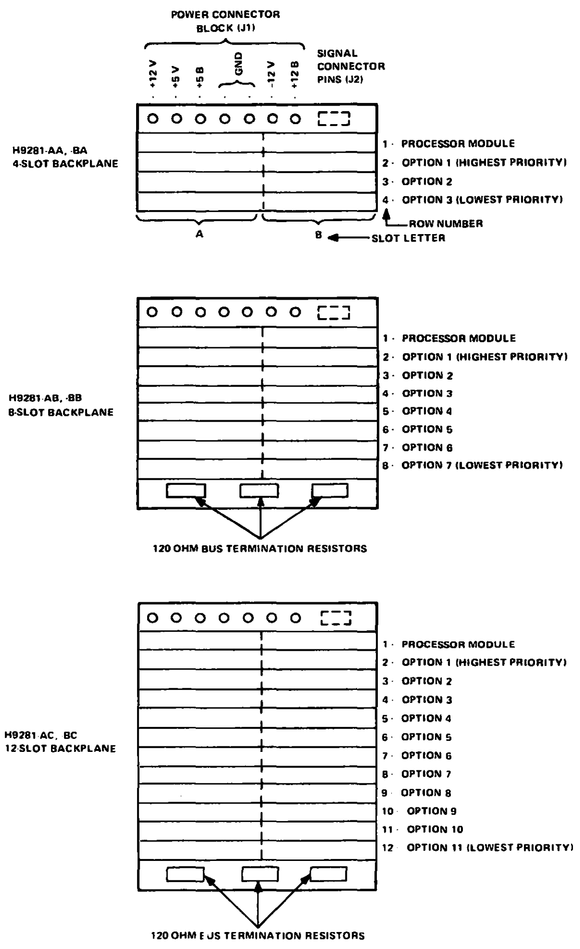


H9273-A Option Positions

When inserted, jumpers W2 and W3 allow the LSI-11 quad-height CPU to run in row 1. Jumpers W2 and W3 are removed when the backplane is used as an expansion backplane in a system.

The connectors designated "Connector 1" are wired according to the LSI-11 bus specification. Slots A and B carry the LSI-11 bus signals and are termed the LSI-11 bus slots. The connectors designated "Connector 2" are wired for +5 V and ground, and have no connections to the LSI-11 bus; instead, C- and D-slot pins on side 2 of each row are connected to the C- and D-slot pins on side 1 in the next lower row.

H9281 Backplane — The H9281 backplanes are designed to accept double-height modules only. The H9281 2-slot backplane is available in six options as listed below. These backplanes allow the user to configure compact LSI-11 bus systems that most efficiently utilize available system space.


POWER CONNECTOR BLOCK (J1)
+12 V
+5 V
+5 B
GND
-12 V
+12 B
SIGNAL CONNECTOR PINS (J2)
H9281-AA, -BA 4-SLOT BACKPLANE
1 - PROCESSOR MODULE
2 - OPTION 1 (HIGHEST PRIORITY)
3 - OPTION 2
4 - OPTION 3 (LOWEST PRIORITY)
ROW NUMBER
A
B
SLOT LETTER
H9281-AB, -BB 8-SLOT BACKPLANE
1 - PROCESSOR MODULE
2 - OPTION 1 (HIGHEST PRIORITY)
3 - OPTION 2
4 - OPTION 3
5 - OPTION 4
6 - OPTION 5
7 - OPTION 6
8 - OPTION 7 (LOWEST PRIORITY)
120 OHM BUS TERMINATION RESISTORS
H9281-AC, BC 12-SLOT BACKPLANE
1 - PROCESSOR MODULE
2 - OPTION 1 (HIGHEST PRIORITY)
3 - OPTION 2
4 - OPTION 3
5 - OPTION 4
6 - OPTION 5
7 - OPTION 6
8 - OPTION 7
9 - OPTION 8
10 - OPTION 9
11 - OPTION 10
12 - OPTION 11 (LOWEST PRIORITY)
120 OHM BUS TERMINATION RESISTORS

H9281 Option and Connector Locations (Module Side)

Backplane Option Designation	**Description**
H9281-AA	4-module backplane
H9281-AB	8-module backplane
H9281-AC	12-module backplane
H9281-BA	4-module backplane and card cage assembly
H9281-BB	8-module backplane and card cage assembly
H9281-BC	12-module backplane and card cage assembly

NOTE

Some options are too large to be installed in an H9281 backplane.

Bus Terminations

Backplane models H9281-AB, -BB, -AC, and -BC include 120 ohm bus termination resistors at the electrical end of the bus; therefore, it is not necessary to install a separate 120 ohm bus terminator module in these backplanes.

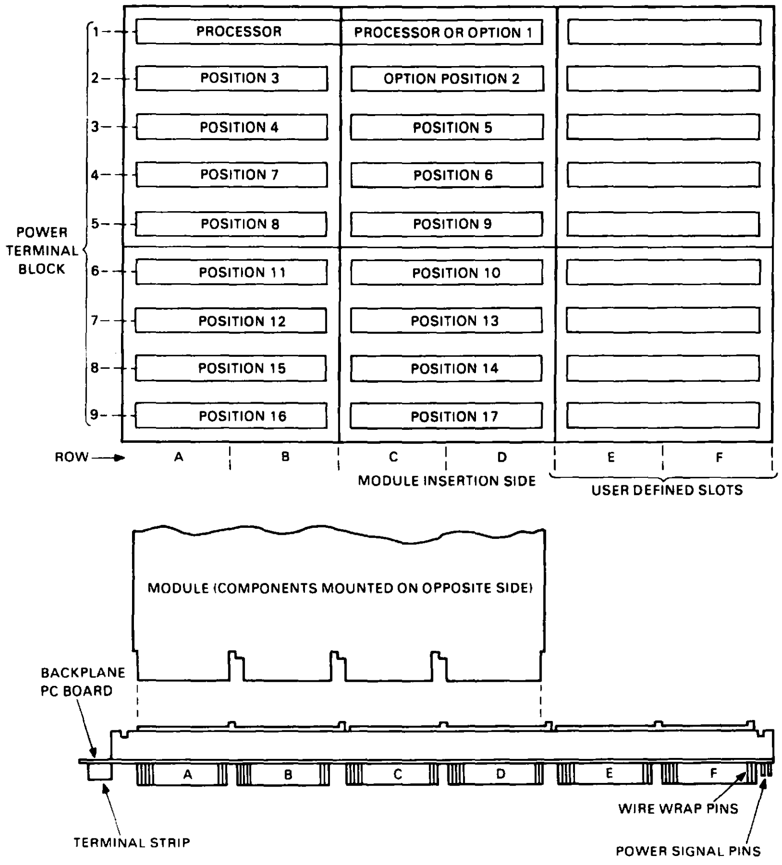
DDV11-B Backplane — The DDV11-B is an optional LSI-11 bus expansion backplane for use when additional logic space is required. The DDV11-B is a 9 × 6, 54-slot backplane with a 9 × 4 slot section (18 individual double-height or 9 quad-height module slots) prebused specifically for LSI-11 bus signal and power and ground connections. The remaining 9 × 2 slot section is provided with +5 Vdc, GND, and −12 Vdc power connections only; this leaves the remaining pins free for use with any special double-height logic modules to be used in conjunction with the LSI-11 family of modules and bus requirements.

Module Slot Assignments

The slot location assignments of the DDV11-B are illustrated in the accompanying figure. Rows A, B, C, and D are dedicated to the LSI-11 bus. Any module that conforms to the LSI-11 bus specifications may be used in this portion of the DDV11-B. The position numbers indicate the bus grant wiring scheme with respect to the processor module. The bus grant signals propagate through the slot locations in the position order shown in the figure below until they reach the requesting device. To provide bus grant signal continuity, any unused slots must be jumpered or unused locations must occur only in the highest position-numbered locations.

Rows E and F contain the 18 user-defined slots with power and ground connections provided.

Device Priority Within Backplanes — All LSI-11 bus backplanes are priority-structured. Daisy-chained grant signals for DMA and interrupt requests propagate away from the processor from the first (highest priority device) to successively lower priority devices.



DDV11-B Module Installation and Slot Assignments

Power Supplies

Both the H780 and the H786 power supplies can be used when configuring a LSI-11/23 system. The H786 is not available separately, only as part of the BA11-N enclosure.

Enclosures

The BA11-M mounting box, which includes an H9270 backplane and an H780 power supply, or the BA11-N mounting box, which includes an H9273 backplane and an H786 power supply, can be used in a system with the LSI-11/23 processor.

Memory Modules

Several memory modules are available for use with the PDP-11/23 systems. However, modules such as MSV11-C or MSV11-D that perform memory refresh locally are required, since the LSI-11/23 does not perform memory refresh itself. MSV11-C memories will work if provision is made for refresh with some other bus option such as REV11; however, this will degrade system performance and is not recommended.

Peripheral Options

All LSI-11 bus-compatible peripheral devices may be used in PDP-11/23 systems. DMA peripherals should be installed with the faster throughput devices physically closest to the processor and slower ones farther away. You must insure that faster devices have adequate access to the bus; otherwise, data drop errors may occur.

Interrupt-driven peripherals can be installed in one of the following ways. If all peripherals use the single-level scheme, they must be installed with faster interrupting devices physically closest to the processor. All current DIGITAL LSI-11 bus peripheral devices must use this method. Future peripheral devices, or customer-designed devices, can take advantage of the new 4-level interrupt scheme. With this scheme, peripherals that are designed to perform distributed interrupt arbitration, and that are on different interrupt levels, can be installed in any order. Multiple peripherals on the same request level and peripherals that do not perform distributed arbitration must be installed with the highest priority, or faster, devices closest to the processor.

SYSTEM DIFFERENCES

A number of minor differences exist between the LSI-11/23 (KDF11-AA) processor and the LSI-11 (KD11-F) or LSI-11/2 (KD11-HA) processor. The following is a list of system differences that exist due to the LSI-11/23's advanced design.

LSI-11/23 has no boot loader in microcode.

Console ODT functions are different in the LSI-11/23.

LSI-11/23 does not perform memory refresh.

The EVENT line is on level 6 in LSI-11/23; LSI-11 and LSI-11/2 have it on level 4.

In systems that used the LSI-11, the ODT command "L" could be used to automatically enter the bootstrap loader. Console ODT in the LSI-11/23 does not contain a bootstrap loader command. Users who are down-line loading to LSI-11/23s must change their host software to enter the 14 memory-word bootstrap loader via console ODT. The

REV11 refresh/boot module cannot be used to boot a LSI-11/23 system. However, the refresh portion of the REV11 can be used to perform refresh for older MSV11-B type memories. This will cause a degradation of system performance and is not recommended. If this method of refreshing memories is employed, the bootstrap/diagnostic functionality of the REV11 must be disabled be removing/installing the appropriate jumpers. The BDV11 bootstrap/diagnostic module may be employed for automatic bootstrap function. The "L" command in the LSI-11 also automatically sizes memory. LSI-11/23 users whose memory size varies will have to create a program to self-size the system or will have to use console ODT.

For improved performance the LSI-11/23 was designed without memory refresh (as was the LSI-11/2). The newer memories such as MSV11-C and MSV11-D perform refresh locally.

In the LSI-11/23, as in all other multi-level interrupt PDP-11 systems, the event line is on *level 6*. In the LSI-11 it is on *level 4*. Users whose own software locked out the event line by just setting PS <07:05> to 4 (priority level 4) will have to modify their software to set PS<07:05> to 6 (priority level 6) when installing a LSI-11/23 into their present system. DIGITAL software is unaffected.

MODULE INSTALLATION PROCEDURE

Follow the procedures listed below:

1. Insure that there is no dc power applied to the backplane.
2. Remove all modules from the backplane.
3. It is recommended that a single switch be used to apply +5 V and +12 V to the backplane. Simultaneous application of +5 V and +12 V is recommended.
4. Turn power on.
5. At the backplane, check for the following voltages with respect to GND (pin C2 in any backplane slot):

   Row 1, Slot A, Pin A2: +5 V
   Row 1, Slot A, Pin D2: +12 V
   Row 1, Slot A, Pin V1: +5 V

   CAUTION

   Do not plug in modules with power applied to backplane.

6. Turn off power.
7. Insure that the system is properly configured.
8. Insert module into backplane.

9. Turn on system power. Observe that the console device responds as described earlier.
10. If the BDV-11 is used as a system bootstrap/diagnostic device, you must consider the following:
    a. The diagnostic portion of the BDV-11 will exercise most legal PDP-11 basic instructions at least once.
    b. The diagnostics were originally created for the LSI-11. In the LSI-11/23 the BDV-11 diagnostics will not:
       (1) perform any memory management or floating point-related tests, or
       (2) exercise any memory present above 32K words.
11. Significant differences exist between console ODT responses generated by the LSI-11 and the LSI-11/23.
12. As a quick check of proper system operation, the following short exerciser program can be used. It prints a continuous stream of ASCII characters on the terminal. Use console ODT to enter the following program.

```
Location    Data        Macro Code
1000        005000            CLR R0
1002        12701             MOV #177564, R1
1004        177564
1006        105711      LOOP: TSTB (R1)
1010        100376            BPL LOOP
1012        110061            MOVB R0, 2 (R1)
1014        2
1016        005200            INC R0
1020        000137            JMP @#1006
1022        001006
```

Enter "1000 G" to console ODT and a continuous stream of ASCII characters should be printed on the terminal.

13. For a more thorough check of the LSI-11/23, processor diagnostics are available to do the following:
    — exercise the basic instruction set
    — exercise the traps and interrupts
    — exercise the memory management and extended addressing functions
    — exercise the floating point hardware registers and the floating instruction set.

The diagnostics are as follows:

- Basic Instruction Set, EIS, Traps and Interrupts Test—CJKDBA
- MMU Diagnostic—CJKDAA
- Floating Point Tests
  Test 1—CJKDCA
  Test 2—CJKDDA

Console Power-Up Printout (or Display)

Conditions	Mode 0	Mode 1	Mode 2	Mode 3
BHALT L (unasserted)	Processor will execute program using contents of location 24 as the PC value.	Terminal will print out a random 6-digit number, which is the contents of the program counter.	Processor will execute program at location 173000. (See Note 2.)	No printout at terminal. (See Note 1.)
BHALT L (asserted)	Terminal will print out contents of memory location 024.	Terminal will print out a random 6-digit number, which is the contents of the program counter.	Terminal will print out "173000." (See Note 2.)	No printout at terminal. (See Note 1.)

NOTES

1. If mode 3 is selected, and user microcode is not implemented, the processor will trap to memory location 10 and start program execution using the contents of location 10 as the PC value and location 12 as the PS value.
2. Normal mode for use with the BDV-11 option. If jumpers W15 through W9 are used, that address will be printed.
3. The terminal printout will consist of 6 octal digits as specifed in the table, followed by a carriage return, line feed, and "@" prompt character in all cases.

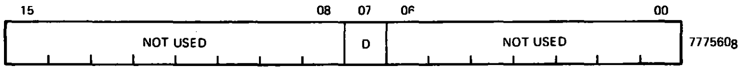
Console octal debugging technique (ODT) exists as a portion of the processor microcode that allows the processor to respond to commands and information entered via the terminal. The terminal addresses are 777560_8 through 777566_8. They are generated in microcode and cannot be changed. Console ODT is useful as an aid in running and debugging programs. Communication between the user and processor is via a stream of ASCII characters interpreted by the processor as console commands. These commands are a subset of ODT-11. The differences in use of console ODT are listed in Appendix G.

Terminal Interface

The minimum hardware requirements for a serial line interface permitting a terminal to communicate with console ODT are contained in the following paragraphs. The intent is to describe the minimum hardware for users who design their own serial line interface. The necessary console ODT hardware is a subset of that needed to operate system software. For system software/hardware requirements refer to the DLV11 section in the **Microcomputer Interface Handbook** of the Microcomputer Handbook Series.

Receiver Control and Status Register (RCSR)

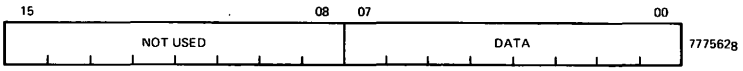
The RCSR must exist at address 777560_8 for character input to console ODT. Console ODT does not execute *DATO* bus cycles to this address; therefore, the RCSR only needs to respond to *DATI* bus cycles. However, system software causes DATO cycles in order to affect certain bits, such as Interrupt Enable (bit 6), which console ODT does not use.



Receiver Status Register

Bit	Description
<7>	Done flag. After a character is assembled and exists in the receiver buffer register (RBUF), the Done flag must be set to a 1. When a DATI is performed to the RBUF (i.e., to pick up the character), the Done flag must be cleared by hardware. Also bus signal BINITL must clear this bit.
<6:0> <15:8>	Unused. These bits are "don't care bits" and can be in any state since console ODT mode does not use them. In DIGITAL interfaces, these bits may be defined.

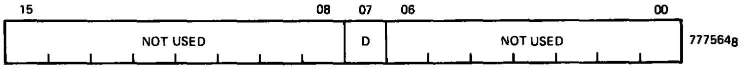
Receiver Buffer Register (RBUF)
The RBUF must exist at address 777562_8 for character input to console ODT. This register only needs to respond to *DATI* bus cycles since console ODT does not execute DATO bus cycles to this address. System software interfaces similarly but DIGITAL diagnostics may cause a DATO cycle and not operate properly.



Receiver Buffer Register

Bit	Description
<7:0>	ASCII character. These eight bits are read by the processor and interpreted as a console ODT command. When bit 7 of RCSR is a 1, the processor does a DATI to the RBUF. After the DATI, the hardware must clear bit 7 of RCSR to 0.
<15:8>	Unused. These bits are "don't care bits" and can be in any state since console ODT does not use them. In DIGITAL interfaces, these bits may be defined.

Transmitter Control and Status Register (XCSR)
The XCSR must exist at address 777564_8 for character output from console ODT. ODT does not execute DATO bus cycles to this address; therefore, the XCSR only needs to respond to DATI bus cycles. However, system software causes DATO cycles to affect certain bits (e.g., Interrupt Enable).

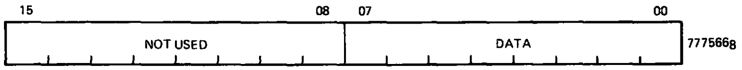


Transmitter Control and Status Register

Bit	**Description**
<7>	Done flag. In the idle state, this bit is a 1, indicating that the hardware is ready to print a character. After a DATO to the transmitter buffer register by the processor (i.e., a character loaded), this bit must be cleared to 0 by the hardware. After the character is printed, the hardware sets this bit to 1. During power-up this bit is set to 1. Bus signal BINIT L must set this bit to 1.
<6:0> <15:8>	Unused. These bits are "don't care bits" and can be in any state since console ODT mode does not use them. In DIGITAL interfaces, these bits may be defined.

Transmitter Buffer Register (XBUF)

The XBUF must exist at address 777566$_8$ for character output from console ODT. This register only needs to respond to DATO bus cycles since console ODT does not execute DATI bus cycles to this address. System software interfaces similarly but DIGITAL diagnostics may cause a DATI cycle and not operate properly.



Transmitter Buffer Register

Bit	**Description**
<7:0>	ASCII character. These eight bits are written by the processor with the ASCII character to be printed. When bit 7 of XCSR is a 1, the processor does a DATO to the XBUF. After the DATO, the hardware must clear bit 7 of XCSR to 0.
<15:8>	Unused. These bits are "don't care" bits and can be in any state since console ODT does not use them. In DIGITAL interfaces, these bits may be defined.

CONSOLE ODT

The processor's microcode operates the serial line interface in half-duplex mode. Program I/O techniques are used rather than interrupts. When the console ODT microcode is busy printing characters using the transmit side of the interface, the microcode is not monitoring the receive side for incoming characters. Any characters coming in at this time are lost. The interface may post overrun errors, but the microcode does not check for any error bit in the interface. Therefore users should not type ahead to ODT because those characters are not recognized. In addition, if another processor is at the other end of the interface, it must obey half-duplex operation. No input characters should be sent until console ODT has finished outputting.

Console ODT Entry Conditions

1. Execution of a HALT instruction in kernel mode, provided the HALT TRAP jumper is not installed.
2. Assertion of the BHALT L signal on the LSI-11 bus. BHALT L is a level, not edge-triggered. The signal must be asserted long enough so that it is seen at the end of a macroinstruction by the service state in the processor.
3. If option 1 has been selected, ODT is entered upon power-up.

NOTE

Unlike the LSI-11 and LSI-11/2, the LSI-11/23 does not enter console ODT upon occurrence of a double bus error (i.e., R6 points to nonexistent memory during a bus timeout trap). The LSI-11/23 creates a new stack at location 2 and continues to trap to 4. Since the LSI-11/23 does not perform memory refresh, a bus timeout during refresh cannot take place. This differs from the LSI-11, which enters console ODT upon such an occurence. If a bus timeout occurs while getting an interrupt vector, the LSI-11/23 ignores it and continues execution of the program, whereas the LSI-11 and LSI-11/2 enter console ODT.

Console ODT Input Sequence

Upon entry to console ODT, the RBUF register is read using a DATI and the character present in the buffer is ignored. This is done so that erroneous characters or user program characters are not interpreted by console ODT as a command, especially when a program is halted.

The input sequence for console ODT is as follows:

1. Read and ignore character in RBUF.
2. Output a <CR><LF> to terminal.
3. Output contents of PC (program counter R7) in six digits to terminal.
4. Output a <CR><LF> to terminal.
5. Output the prompt character, @, to terminal.
6. Enter a wait loop for terminal input. The Done flag, bit 7 in RCSR, is tested using a DATI. If it is 0, the test continues.
7. If RCSR bit 7 is a 1, then low byte of RBUF is read using a DATI.

Console ODT Output Sequence

The output sequence for ODT is as follows:

1. Test XCSR byte 7 (Done flag) using a DATI and if a 0, continue testing.
2. If XCSR bit 7 is 1, write character to low byte of XBUF using a DATO (high byte is ignored by interface).

CONSOLE ODT COMMAND SET

The console ODT terminal set, listed below, is described in the following paragraphs. The commands are a subset of ODT-11 and use the same command character. Console ODT has ten internal states. For each state only specific characters are recognized as valid inputs; other inputs invoke a "?" response. These states are described below.

Console ODT Commands

Command	**Symbol**	**Use**
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 an $ or R command.

Command	Symbol	Use
Go	G	Starts program execution.
Proceed	P	Resumes execution of a program.
Binary Dump	Control Shifts-S	Manufacturing use only.
	H	Reserved for DIGITAL use.

Console ODT States and Valid Input Characters

State	Example of Terminal Output	Valid Input	Comment
1	@	0-7 R,S G P Control-Shift-S	
2	@R or @$	0-7 S	
3	@1000/ 123456	0-7 CR LF	
4	@R1/123456	0-7 CR LF	
5	@1000	0-7 / G	
6	@RI or @RS	0-7 S /	
7	@1000/ 123456 1000	0-7 CR LF	

State	Example of	Valid Input	Comment
8	@R1/ 123456 1000	0-7 CR LF	
9	@	/	Previous location was opened
10	@ Control-Shift-S	2 binary bytes	

The parity bit (bit 7) on all input characters is ignored (i.e., not stripped) by console ODT and if the input character is echoed, the state of the parity bit is copied to the output buffer (XBUF). Output characters internally generated (e.g., <CR>) by ODT have the parity bit equal to 0. All commands are echoed except for <LF>. Where applicable, uppercase and lowercase command characters are recognized.

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

NOTE

In the examples, the response from the processor is underlined, while the user's entry is not.

/(ASCII 057) Slash

This command is used to open an LSI-11 bus address, processor register, or processor status word and is normally preceded by other characters which specify a location. In response to /, console ODT prints the contents of the location (i.e., six characters) and then a space (ASCII 40). After printing is complete, console ODT waits for either new data for that location or a valid close command. The space character is issued so that the location's, contents and possible new contents entered by the user are legible on the terminal.

Example: @001000/012525<SPACE>

where:

@	= console ODT prompt character.
001000	= octal location in the LSI-11 bus address space desired by the user (leading 0s mare not required).
/	= command to open and print contents of location.
012525	= contents of octal location 1000.
<SPACE>	= space character generated by console ODT.

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

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

where:

first line	= new data of 1234 entered into location 1000 and location closed with <CR>
second line	= a / was entered without a location specifier and the previous location was opened to reveal that the new contents were correctly entered into memory.

<CR>(ASCII 15) Carriage Return

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

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

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

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

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

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

<LF> (ASCII 12) Line Feed
This command is used to close an open location and then open the next contiguous location. LSI-11 bus addresses and processor registers are incremented by 2 and 1 respectively. If the PS is open when a <LF> is issued, it is closed and a <CR><LF>@ is printed; no new location is opened. If the open location's contents are to be changed, the new data should precede the <LF>. If no data is entered, the location is closed without being altered.

Example:

```
@R2/123456<SPACE> <LF> <CR><LF>
@R3/054321<SPACE>
```

In this case, the user entered <LF> with no data preceding it. In response, console ODT closed R2 and then opened R3. When a user has the last register, R7, open, and issues <LF>, console ODT opens the beginning register, R0. When the user has the last LSI-11 bus address open of a 32K word segment and issues <LF>, console ODT opens the first location of that same segment. If the user wishes to cross the 32K word boundary, he must reenter the address for the desired 32K word segment (i.e., console ODT is module 32K word). This operation is the same as that found on all other PDP-11 consoles.

Example:

```
@R7/000000<SPACE> <LF> <CR><LF>
@R0/123456<SPACE>
```

or

```
@577776/000001<SPACE> <LF> <CR><LF>
@477776/125252<SPACE>
```

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

$ (ASCII 044) or R (ASCII 122) Internal Register Designator
Either character when followed by a register number, 0 to 7, or PS designator, S, will open that specific processor register.

The $ character is recognized to be compatible with ODT-11. The R character was introduced for the convenience of one key stroke and because it is representative of what it does.

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

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

S (ASCII 123) Processor Status Word
This designator is for opening the PS (processor status word) and must be employed after the user has entered an R or $ register designator.

```
Example:    @RS/100377<SPACE> 0 <CR> <CR><LF>
            @/000010<SPACE>
```

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

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

G (ASCII 107) Go
This command is used to start program execution at a location entered immediately before the G. This function is equivalent to the LOAD ADDRESS and START switch sequence on other PDP-11 consoles.

```
Example:    @200 <NULL><NULL>
```

The console ODT sequence for a G, after echoing the command character, is as follows:

1. Print two nulls (ASCII 0) so the LSI-11 bus initialize that follows does not flush the G character from the double-buffered UART chip in the DLV11 serial line interface.
2. Load R7 (PC) with the entered data. If no data is entered, 0 is used. (In the above example, R7 is equal to 200 and that is where program execution begins).
3. The PS, and floating point status register if the MMU is present, is cleared to 0.

4. The LSI-11 bus is initialized by the processor asserting BINIT L for 12.6 microseconds (at 300 ns microcycle), negating BINIT L, and then waiting for 110 microseconds (at 399 ns microcycle).
5. The service state is entered by the processor. If there is anything to be serviced, it is processed. If the BHALT L bus signal is asserted, the processor reenters the console ODT state. This feature is used to initialize a system without starting a program (R7 is altered). If the user wants to single-step his program he issues a G and then successive P commands, all done with the BHALT L bus signal asserted.

P (ASCII 120) Proceed
This command is used to resume execution of a program and corresponds to the CONTINUE switch on other PDP-11 consoles. No programmer-visible machine state is altered using this command.

Example:
<u>@</u>P

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

Control-Shift-S (ASCII 23) Binary Dump
This command is used for manufacturing test purposes and is not a normal user command. It is described here to explain the machine's response if accidentally invoked. It is intended to more efficiently display a portion of memory compared to using the "/" and <LF> commands. The protocol is as follows:

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

   The first byte specifies starting address <15:08> and the second byte specifies starting address <07:00>. Bus address bits <17:16> are always forced to be 0; the dump command is restricted to the first 32K words of address space.
3. After the second address byte has been received, console ODT outputs 12 octal bytes to the serial line starting at the address

previously specified. When the output is finished, console ODT prints <CR><LF>@.

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

Reserved Commands
An ASCII H is reserved for future DIGITAL use. If it is accidentally typed, console ODT will echo the H and print a prompt character rather than a "?" which is the invalid character response. No other operation is performed.

ADDRESS SPECIFICATION

All I/O addresses (24K to 128K) must be entered by users with all 18 bits specified, regardless of whether the MMU is present or not. For example, if a user desires to open the RCSR of the DLV11, he must enter 777560, not 177560. With a MMU present, 18-bit addresses must be used to access memory greater than 32K words.

Processor I/O Addresses
Certain processor and MMU registers have I/O addresses assigned to them for programming purposes. If referenced in console ODT, the PS responds to its bus address, 777776. Processor registers R0 through R7 do not respond (i.e., timeout occurs) to bus addresses 777700 through 777707 if referenced in console ODT.

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

Example: @777572/000001<SPACE>

In this case, memory management status register 0 is opened and the memory management enable is set.

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

Stack Pointer Selection
Similarly, if a program is operating in user mode, "R6" accesses the user stack pointer. If a specific stack pointer is desired, PS<15:14> must be set by the user to the appropriate value and then the "R6"

command can be used. If an operating program has been halted, the original value of PS<15:14> must be restored in order to continue execution.

```
Example:    PS = 140000
            @R6/123456<SPACE>
```

The user mode stack pointer has been opened.

```
            @RS/140000<SPACE> 0<CR> <CR><LF>
            @R6/123456<SPACE> <CR< <CR><LF>
            @RS/000000<SPACE> 140000<CR> <CR><LF>
            @P
```

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

If PS<15:14> is set to 01, another unique register exists in the processor, but is reserved for future DIGITAL use.

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

ENTERING OCTAL DIGITS

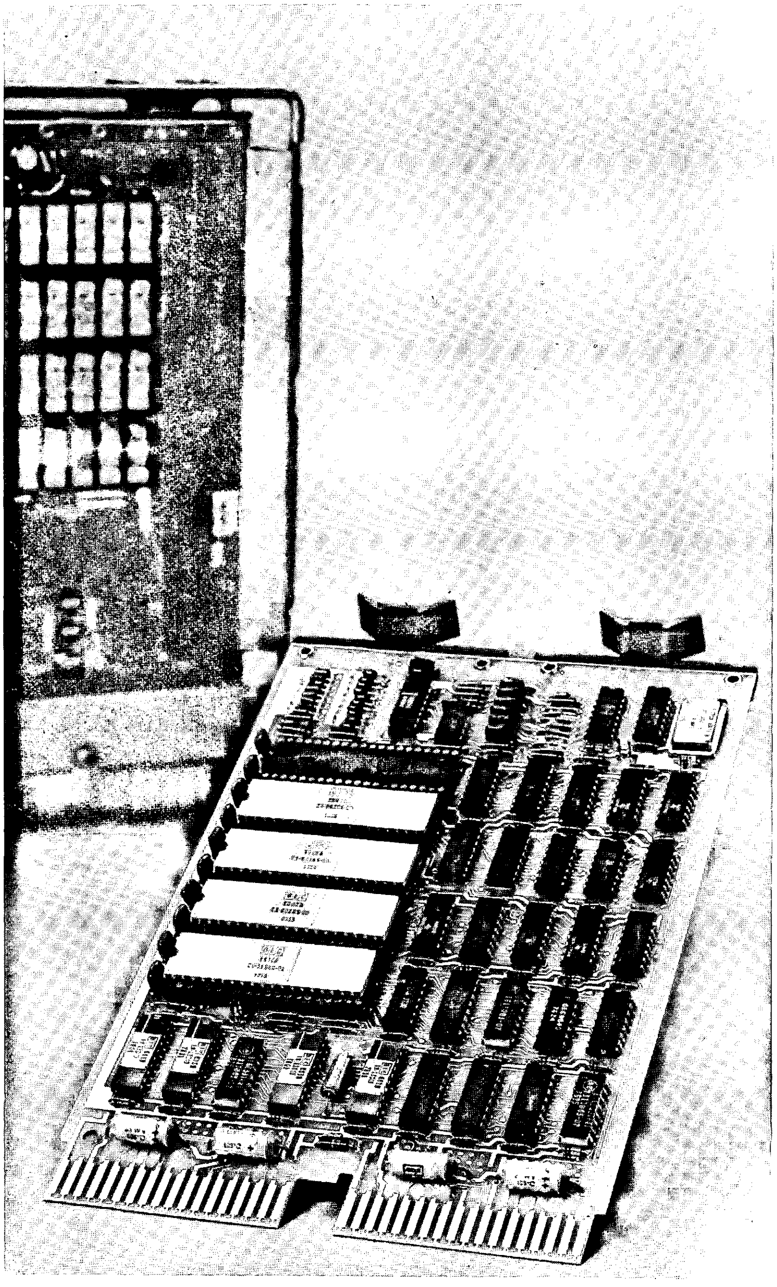
When the user is specifying an address of data, console ODT will use the last six octal digits if more than six have been entered. The user need not enter leading 0s for either address or data; console ODT forces 0s as the default. If an odd address is entered, the low-order bit is ignored and full 16-bit words are displayed.

ODT TIMEOUT

If the user specifies a nonexistent address or causes a parity error, console ODT responds to the error by printing ?<CR><LF>@.

INVALID CHARACTERS

Console ODT will recognize uppercase and lowercase characters as commands. Any character that console ODT does not recognize during a particular sequence is echoed (with the exception of ASCII 0, 2, 10, or 12 as noted earlier) and console ODT prints a ?<CR><LF>@. Console ODT has ten internal states, each of which has its own set of valid input characters. When in a particular state, only commands specific to that state are valid. This was done to lower the probability of a user unintentionally destroying a program by pressing the wrong key.



CHAPTER 4
LSI-11/2 PROCESSOR

GENERAL

The LSI-11/2 is a 16-bit microcomputer with the speed and instruction set of a minicomputer. Due to its size and unique capabilities, it can fit into almost any instrumentation, data processing, or controller configuration.

A complete and powerful microcomputer system can be configured using the LSI-11/2, appropriate memory, I/O devices, and interconnection hardware.

The LSI-11 bus handles all communication between modules and connects the memory and I/O interface elements to the central processor. It contains multiple high-speed, general-purpose registers which can be used as accumulators, address pointers, index registers, and for other specialized functions. The processor does both single- and double-operand addressing and handles both 16-bit word and 8-bit byte data. The bus permits DMA data transfers directly between I/O and memory without disturbing the processor registers.

FEATURES

- Extended Instruction Set (EIS) available as an option.
- Floating Point Instruction Set (FIS) available as an option.
- No on-board memory - flexibility to match RAM/ROM size to requirements.
- Compact, double-height module size for versatile packaging.
- ODT console emulator for ease of program debugging.
- Direct addressing of 32K 16-bit words or 64K 8-bit bytes (K = 1024).
- Over 400 instructions for powerful and convenient programming.
- 16-bit word or 8-bit byte addressable locations.
- Eight internal general-purpose registers for use as accumulators and for operand addressing.
- Stack processing for easy handling of structured data, subroutines, and interrupts.
- Efficient processing of 8-bit characters without the need to rotate, swap, or mask.
- LSI-11 bus structure that provdies position-dependent priority as peripheral device interfaces are connected to the I/O bus.

- Asynchronous bus operation allows processor and system components (memory and peripherals) to run at their highest possible speed.
- Direct memory access (DMA) allows peripherals to access memory without interrupting processor operation.
- Fast interrrupt response without device polling.
- Power fail and automatic restart hardware detect and protect against ac power fluctuations.
- Modular component design allows systems to be easily configured and upgraded.

SPECIFICATIONS

Identification	M7270
Size	Double
Dimensions	13.34 cm × 22.8 cm (5.25 in × 8.9 in)
Power Requirements	+5V ± 5%, 1.0 A +12 V ± 5%, 0.22 A
Bus Loads	ac 1.7 unit loads dc 1 unit loads
Instruction Timing	(See appendix A)
Interrupt Latency	35.05 Microsections ±20% (worst case if KEV11 option not present) 44.1 microseconds ±20% (worst case if KEV11 option is present)
DMA Latency	6.45 microseconds ±20% (worst case)
ENVIRONMENTAL	
Operating Temperature	5° C to 60° C (41° to 140° F) —Derate the maximum temperature by one degree Celcius for each 1000 feet of altitude above 8000 feet.
Relative Humidity:	10% to 90%, non-condensing
Altitude:	Up to 50,000 feet (note temperature derating above 8000 feet.)

Airflow:	Sufficient air flow must be provided to limit the temperature rise across the module to 5°C for an inlet temperature of 60°C. For inlet air temperature below 55°C, air flow must be provided to limit temperature rise across the module to 10°C.

NOTE

These are the design limits. Lower temperature limits will serve to increase the life of the module.

ENVIRONMENTAL

Storage Temperature:	−40°C to 65°C (−40°F to 149°F)
Relative Humidity:	10% to 90%, non-condensing
Altitute:	Up to 50,000 feet

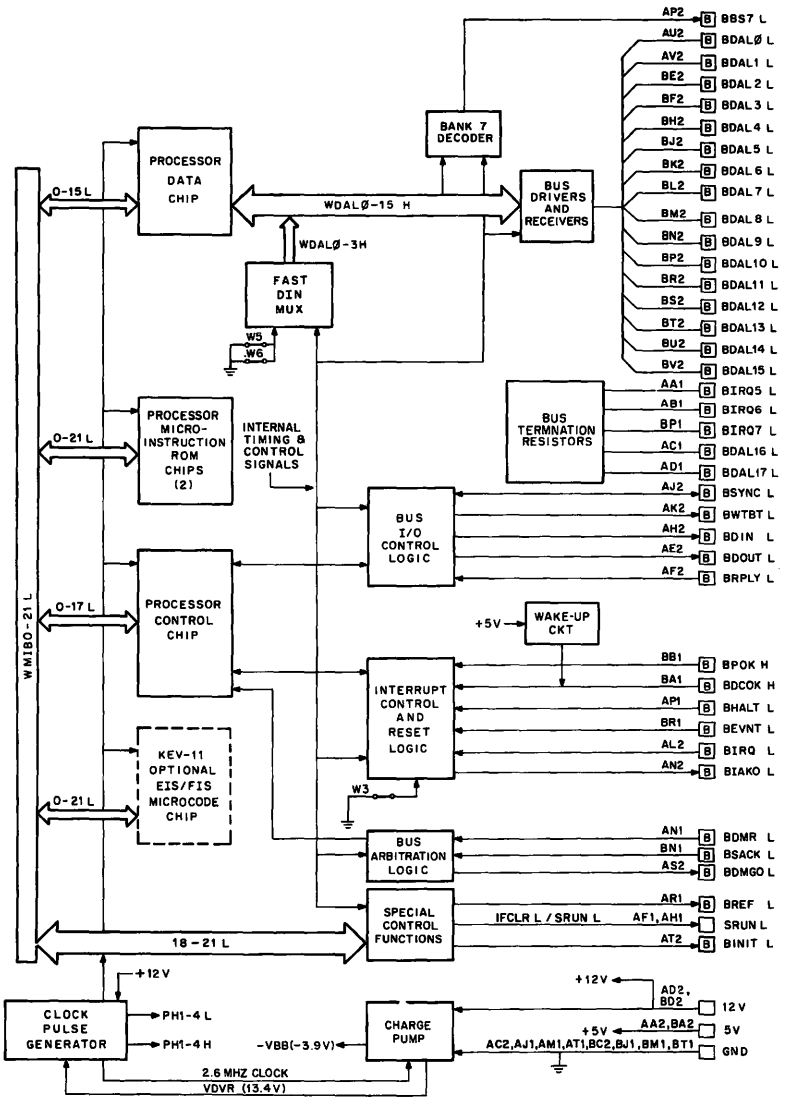
NOTE

When stored outside the operating range, the module should be allowed to stabilize in the operating range for a minimum of 5 minutes before operating.

DESCRIPTION

The LSI-11/2 processor (KD11-HA) is a double-height module, 5 ½″ × 8 ½″ (13.3 cm × 22.8 cm).

Option No.	**Module**	**Description**
KD11-HA	M7270	LSI-11/2 processor module only (no memory)
KD11-HB	M7270 M8044-B	LSI-11/2 processor module plus double-height MSV11-DB 8K × 16-bit read/write memory module
KD11-HC	M7270 M8044-CA	LSI-11/2 processor module plus double-height MSV11-DC 16K × 16-bit read/write memory module
KD11-HD	M7270 M8044-DA	LSI-11/2 processor module plus double-height MSV11-DD 32K × 16-bit read/write memory module
KD11-HF	M7270 M8044-DA	LSI-11/2 processor module plus double-height MMSV11-DA 4K × 16-bit read/write memory module
KD11-HU	M7270 M8021	LSI-11/2 processor module plus double-height MRV11-BA 4K ultraviolet erasable programmable read-only memory (UV PROM) 256-word read/write memory module. Memory module is supplied without UV PROM integrated circuits. Sockets are mounted on the module for user installation of PROM integrated circuits (type MRV11-BC)



M7270 LSI-11/2 Processor Module Basic Functions

CHIP SET

The main functions of the processor module are performed by the microprocessor chip set. The LSI-11/2 chip set includes:

- one control chip
- one data chip
- two microinstruction ROM chips, microms
- one optional KEV11 MICROM with EIS/FIS (Extended Instruction Set/Floating Instruction Set)

The microprocessor chips communicate with each other over a 22-bit **microinstruction bus.** All address and data communication between the microprocessor chips and other processor module functional blocks is via the data chip and the 16 bit data address lines, WDAL <0: 15> H (from the data chip).

Processor module control signals interface with the microprocessor chips via the control chip. Eight input and five output microprocessor control signals provide this function.

Timing and synchronization of all microprocessor chips (and all processor module functions) are controlled by four nonoverlapping clock pulses. Typical operating speed is 380 ns (95 ns each phase).

The control chip generates a sequence of microinstruction addresses that access the microinstruction microm chips. The addressed microinstruction is then transferred to the data and control chips. Most of the microinstructions are executed by the data chip; however, various jumps, branches, and I/O operations are executed in the control chip.

The data chip contains the data paths, logic, arithmetic logic unit (ALU), processor status bits, and registers. Registers include the eight general registers (R0-R7) and an instruction register. The user's program has access to all general registers and processor status (PS) bits. All PDP-11 instructions enter this chip via the WDAL bus. Data and addresses to and from the microprocessor are also transferred to and from the processor over this 16-bit bus.

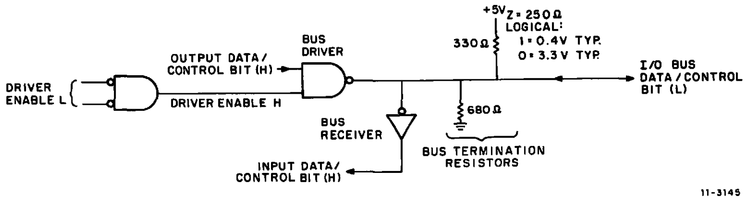
CAUTION

Do not remove processor chips from their sockets. Improper handling will permanetly damage the chips.

Bus Interface and Data/Address Distribution

All LSI-11/2 processor module communication to and from external I/O devices and memories is accomplished using the LSI-11 bus 16-

bit data/address lines (BDAL <0:15> L) and bus control signals. The processor module interfaces to the bus using bus driver/receiver chips, as shown in the LSI-11 Bus Loading and Driver/Receiver Interface figure. Each bus driver/receiver chip contains four open-collector drivers and four high-impedance receivers. Each driver output is common to a receiver input. Either processor output data (from the driver outputs) or input data (from the bus) can stimulate bus receiver inputs.



LSI-11 Bus Loading and Driver/Receiver Interface

All four drivers in a chip are enabled or disabled by a pair of DRIVER ENABLE L inputs. A high input will inhibit all four drivers, when both enable inputs are low, the drivers are enabled and output data is gated onto the bus.

DMGCY H and INIT (1) H are processor module logic control signals that inhibit certain bus drivers during an Initialize or DMA operation. Bus drivers are enabled when these signals are in the false (low) state.

Bus driver output signals and their respective enable signals:

Bus Driver (Signal)	**Enable Signal(s) (Low = Enable)**
BSYNC L	
BBS7 L	INIT (1) H, DMGCY H
BREF L	
BIAKO L	
BWTBT L	
BRPLY L	
BDIN L	INIT (1) H
BDOUT L	
BINIT L	Always enabled
BDMGO L	

The near-end bus termination resistors are contained on the processor module. Each bus driver output is terminated by a pair of resistors, as shown in the figure, establishing the nominal 250Ω bus impedance and the 3.4 V nominal voltage level.

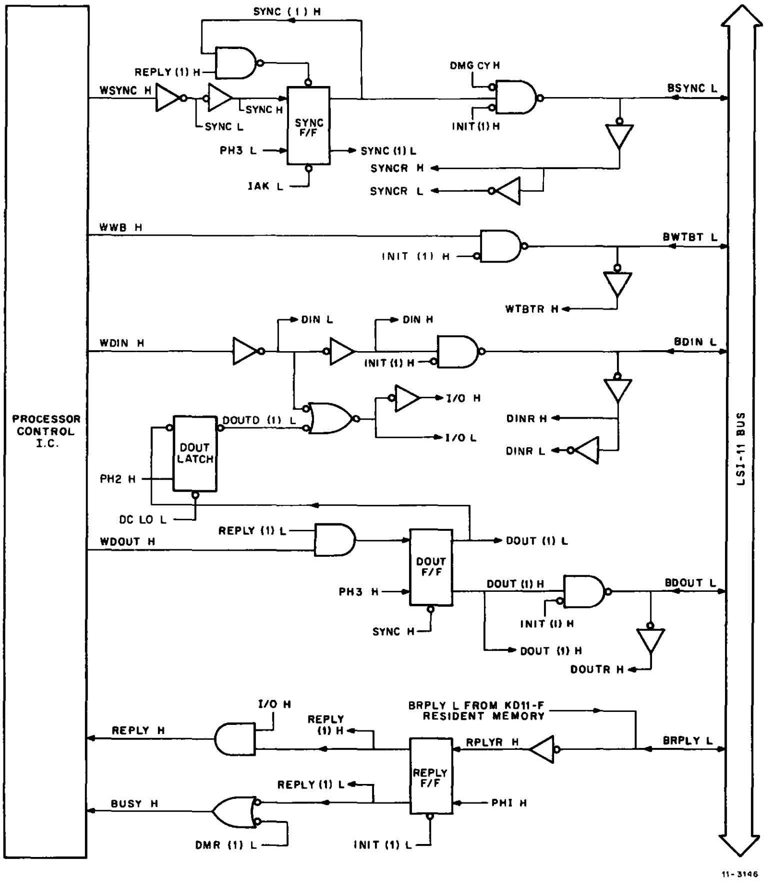
Address and data information are distributed on the processor module via the WDAL <0:15> H and DAL <0:15> H 16-bit buses. WDAL <0:15> H interface directly with the microprocessor data chip, the DEC 8641 bus drivers. All processor input data from the I/O bus is via the bus receivers, the DAL <0:15> H bus, the data multiplexer, the WDAL <0:15> H bus, and the microprocessor data chip.

Bus I/O Control Signal Logic — Bus I/O control signals include BSYNC L, BWTBT L, BDIN L, BDOUT L, and BRPLY L. In addition, BIAKO L can be considered a bus I/O control signal; however, since it is only used during the interrupt sequence, it is discussed later. Logic circuits which produce and/or distribute these signals are shown in the Bus I/O Control Signal Logic. Each signal is generated or received as described in the following paragraphs.

BSYNC L—The control chip initiates the BSYNC L signal sequence by raising WSYNC H during PH2. Inverters apply the high SYNC H signal to the Sync flip-flop sets, producing an active (high) SYNC (1) H input to the BSYNC L bus driver. SYNC (1) H is gated with REPLY (1) H (when active) to produce a direct preset input to the Sync flip-flop. This ensures that BSYNC L will remain active until after the bus slave device terminates its BRPLY L signal and the Reply flip-flop is reset. [REPLY (1) H is low.] The Sync flip-flop then clocks to the reset (BSYNC L passive) state on the trailing edge of PH3 L.

BWTBT L—BWTBT L is the buffered/inverted control chip WWB H output signal. This signal asserts during PH1 of the addressing portion of a bus cycle to indicate that a write (output) operation follows. It remains active during the output data transfer if a DATOB bus cycle is to be executed.

BDIN L—BDIN L is the inverted, buffered control chip WDIN H signal. This signal goes active during PH2 following an active RPLY H signal.



Bus I/O Control Signal Logic

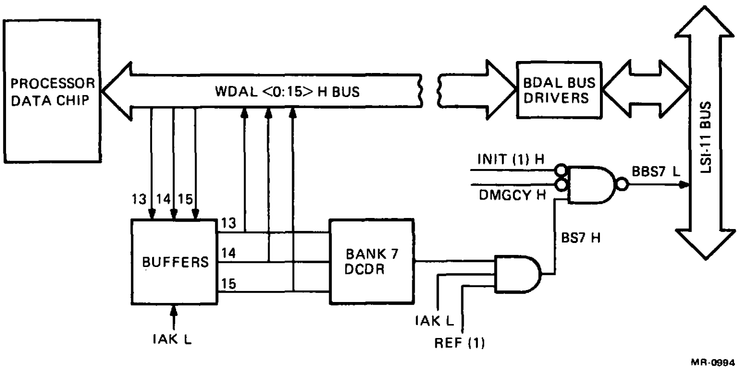
BDOUT L—The control chip initiates the BDOUT L signal sequence by raising WDOUT H during PH2. This signal is gated with the passive REPLY (1) L (high) signal to produce an active (low) D input to the DOUT flip-flop. The flip-flop sets on the leading edge of PH3 H, producing an active BDOUT L signal. It clocks to the reset state on PH3 following the REPLY (1) active (low) signal.

BRPLY L—BRPLY L is a required response from a bus slave device during input or output operations. DIN L and DOUT (1) L are ORed to produce an active I/O signal whenever a programmed transfer occurs.

I/O L enables the time-out counter in the bus error detection portion of the interrupt logic. I/O is inverted to produce I/O H, which enables the reply gate REPLY H signal input to the control chip.

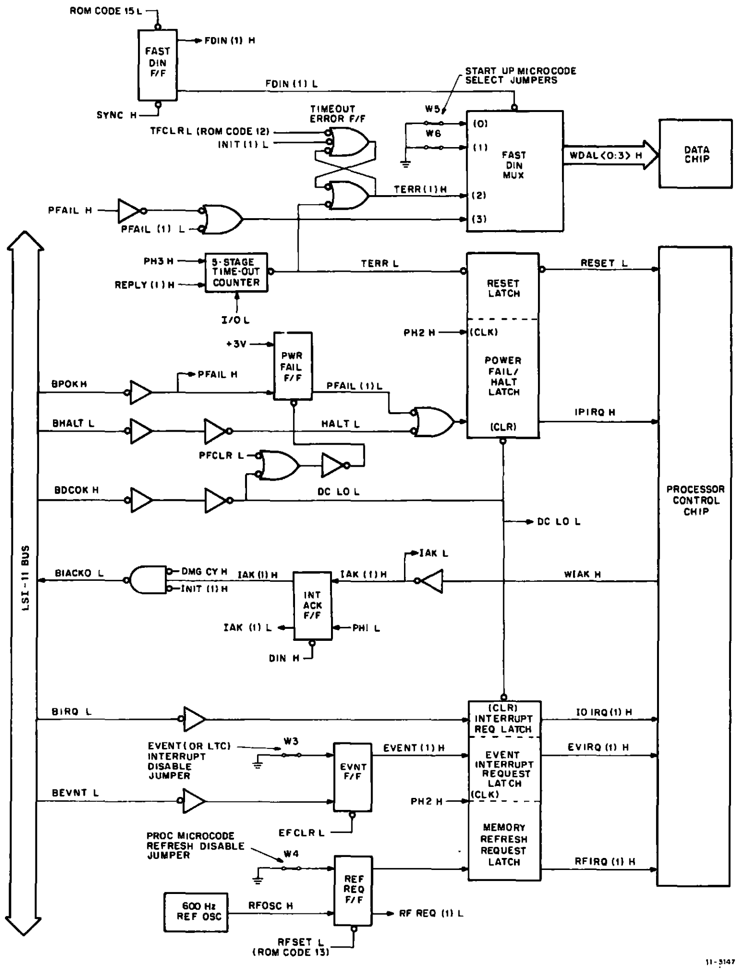
BRPLY L is received either from the LSI-11 bus or resident memory and inverted to produce a high input to the Reply flip-flop. PH1 H clocks the flip-flop to set state, producing active REPLY (1) H and REPLY (1) L signals. REPLY (1) L is ORed with DMR (1) L to produce an active BUSY H signal. The control chip responds by entering a wait state, inhibiting completion of the processor-generated bus transfer for the duration of REPLY (1) L. REPLY (1) H is gated with I/O H to produce an active REPLY H signal, informing the processor that the output data has been taken or that input data is available on the bus. REPLY H goes passive when I/O H goes passive. The bus slave device will then terminate the BRPLY L signal, indicating that it has completed its portion of the data transfer. On the next PH1 H clock pulse, the Reply flip-flop resets and REPLY (1) H and L and BUSY H go passive.

Bank 7 Decoder — The bank 7 decode circuit is shown in the accompanying figure. Buffers receive WDAL <0:15> H bits and distribute them to the bank 7 decoder and BDAL bus drivers. Bank 7 is decoded during the addressing portion of the bus cycle. If a peripheral device address is referenced, an address in bank 7 (28—32K address space) is used, and WDAL <13:15> H are all active (high). This address is decoded and BBS7 L is asserted. When active, BBS7 L enables addressing of non-memory devices along the bus. During interrupt vector bus transactions, IAK L becomes asserted. IAK L inhibits BS7 H and BBS7 L generation, which could result in an invalid input data transfer. REF(1) L inhibits BS7 H and BBS7 L generation during memory refresh bus cycles.



Bank 7 Decoder

Interrupt Control and Reset Logic — Interrupt control and reset logic functions are shown in the accompanying figure. Reset functions include bus error and power-fail (BDCOK H negated). Interrupt functions include power-fail (impending), Halt mode (console microcode control), refresh interrupt, event (or line time clock) interrupt, and external BIRQ interrupts.



Interrupt Control and Reset Logic

Power-Fail/Restart Sequence — A power-fail sequence is initiated when BPOK H goes low, clocking the Power-Fail flip-flop to the set state. PFAIL (1) L is ORed with HALT L to produce a high signal. This signal is latched during PH2 H, producing an active IPIRQ H (interrupt 1) input to the control chip. The processor then interrupts program execution. Note that the low (passive) BPOK H signal is inverted to produce an active PFAIL H input to the fast DIN multiplexer; the signal status is checked by the microcode to ensure that BPOK H is asserted.

Upon entry to this microcode routine, the processor requests a fast DIN cycle. This request is decoded as ROM CODE 15 L, presetting the fast DIN flip-flop. FDIN (0) H goes low, enabling the fast DIN multiplexer to place power-up mode option jumper data, the passive time-out error [TERR (1) H] signal, and the active PFAIL H signal on WDAL <0: 3> H. The processor receives the fast DIN information via the data chip. An active PFAIL H signal informs the processor that a power-fail condition is in progress, rather than the halt condition.

BDCOK H goes passive (low) and produces an active DC LO L signal, clearing the Power-Fail flip-flop and the power-fail/halt and reset latches and initializing the processor and all devices. The active RESET L signal then initializes the processor, causing it to abort console (halt) or power-fail microcode execution and enter a "no operation" state. The processor remains in this condition until BDCOK H returns to the active state.

Once initiated, the power-fail sequence must be completed before the power-up sequence is started, otherwise the processor will "hang."

The power-up restart condition occurs when DC LO L goes false; RESET L goes passive (high) on the next PH2 H clock pulse. The processor responds by executing a fast DIN cycle to determine the start-up microcode option jumper configuration. Once the fast DIN cycle has been completed, the processor executes the power-up option selected, and normal operation resumes when BPOK H is asserted.

Halt Mode — The Halt mode is entered by executing the HALT instruction, by a device asserting the BHALT L signal, by a double bus error condition, or by a bus error (time-out) during an interrupt. The processor halts program execution and enters microcode execution as described for a power-fail operation. However, when the processor executes the fast DIN cycle, the PFAIL H bit (WDAL3 H) is not active and console microcode (not a power-fail sequence) is executed. Negation of BHALT L will allow the processor to resume PDP-11 program execution. On the next PH2 H clock pulse, IPIRQ H goes false (low) and the processor Run mode is enabled.

Bus Errors — A bus error results in aborting program execution and entry into a trap service routine via vector location 004. A bus error occurs when a device fails to respond to the processor DBIN L or DBOUT L signal by not returning a BRPLY L signal within 10 μs (approximately). An active I/O signal inhibits the reset input of the 5-stage time-out counter, enabling counter operation. [When not in a processor-controlled bus I/O cycle, I/O L is passive (high), clearing the counter.] The counter proceeds with counting PH3 H clock pulse signals. Normally BRPLY L would be asserted, producing an active REPLY (1) H signal which inhibits the counter; the count would remain stable until cleared by a passive I/O L signal. However, if BRPLY L is not received within 10 μs, the full count (32_10) is attained. This is the error condition; TERR L goes low and TERR (1) H goes high. The next PH2 H clock pulse clocks the reset latch to the reset (active) state, producing an active RESET L signal. The processor responds by executing the reset microcode. After entering the microcode, the processor executes a fast DIN cycle and determines that a time-out (bus) error TERR (1) H, rather than a power-fail condition, has occurred. It then responds by executing the bus error trap service routine. TFCLR L (ROM code 2) is generated by the processor to clear the TERR latch.

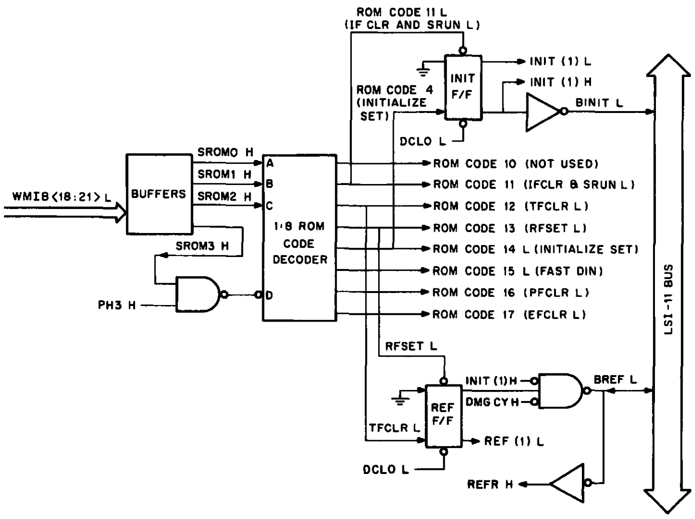
Normal I/O Interrupts — "Normal" I/O interrupts are those interrupt requests that are generated by external devices using bus interrupt request BIRQ L. The request is initiated by asserting BIRQ L. This signal is inverted to produce a high signal, which is stored in the interrupt request latch on the next PH2 H pulse. The stored request produces IOIRQ (1) H, which informs the processor of the request. If processor status word priority is 0, the processor responds by producing an active WIAK H (interrupt acknowledge) and WDIN H signals. WDIN H is buffered onto the BDIN L signal line to signal devices to stabilize their priority arbitration. WIAK H is inverted, producing IAK L, setting the Interrupt Acknowledge flip-flop on the trailing edge of PH1 L one cycle after BDIN L is asserted. The high (active) interrupt acknowledge signal is enabled onto the BIAKO L signal line by passive (low) DMGCY H and INIT (1) H signals. The highest priority device requesting interrupt service responds to the processor BDIN L and BIAK L signals by placing its vector on the BDAL bus and asserting BRPLY L, inputting its vector to the processor of the request. Note that BSYNC L is not asserted during this operation and that no device addressing occurs. The device also clears its BIRQ L signal. The processor responds to BRPLY L by terminating BDIN L and BIAK L.

Refresh — Memory refresh is initiated by a 600 Hz refresh oscillator. This function is enabled when jumper W4 is not installed. The leading edge of RFOSC H clocks the Refresh Request flip-flop to the set state.

On the next PH2 H clock pulse, the memory refresh request latch stores the request and applies an active RFIRQ H signal to the processor control chip. The processor responds by producing an active RF SET L signal and executing the refresh microcode. RF SET L sets the Refresh flip-flop, producing the BREF L signal and clearing the Refresh Request flip-flop, which terminates the request. TFCLR L resets the Refresh flip-flop when the refresh operation is completed. Note that BREF L is not asserted if DMGCY H or INIT (1) H is asserted.

Event Line Interrupt — The event line interrupt function can be used as a line time clock interrupt, or as desired by the user. This interrupt differs from the normal I/O interrupt request by being the highest priority external interrupt, and it does not input a vector in order to enter its service routine. The interrupt is initiated by the external device by asserting BEVNT L. This signal is inverted to produce a high (active) signal, which clocks the Event flip-flop to the set state. (Note that when W3 is installed, the flip-flop remains reset and the event function is disabled.) On the next PH2 H clock pulse, the event interrupt request latch stores the active EVNT (1) H signal. An active EVIRQ (1) H signal is then applied to the control chip. If processor status word priority is 0, the interrupt will be serviced. Service is gained via vector 100_8< which is dedicated to the event interrupt. Hence, a bus DIN operation does not occur when obtaining the vector. The request is cleared by the microcode-generated EFCLR L signal.

Special Control Function — Special control functions include microcode-generated bus initialize and memory refresh operations and five special control signals which are internal to the processor module. Special control function logic circuits are shown in the accompanying figure. Microinstruction bus lines WMIB <18:21> L are buffered to proudce the four SROM <0:3> H signals. The actual codes for the special functions are contained on SROM <0:2> H; SROM3 H is always active when a special function is to be decoded, enabling the 1: 8 ROM code decoder during PH3 H. The resulting decoded functions are described below.



Special Control Functions

ROM Code 10 — Not used.

ROM Code 11 [IFCLR and SRUN L] — This code is produced by the processor to clear the initialize flip-flop and to assert the SRUN L signal for an external RUN indicator circuit.

ROM Code 12 [TFCLR L] — This code is a trap function clear signal which clears the Refresh Request and Time-Out Error flip-flops.

ROM Code 13 [RFSET L] — This code is used to set the Refresh flip-flop. The active (high) flip-flop output is gated with passive (low) INIT (1) H and DMG (1) H signals to produce the active BREF L signal. The flip-flop normally resets by the microcode-generated TFCLR L signal after completing the refresh operation, or whenever a power failure occurs. (DC LO L goes active and clears the flip-flop.)

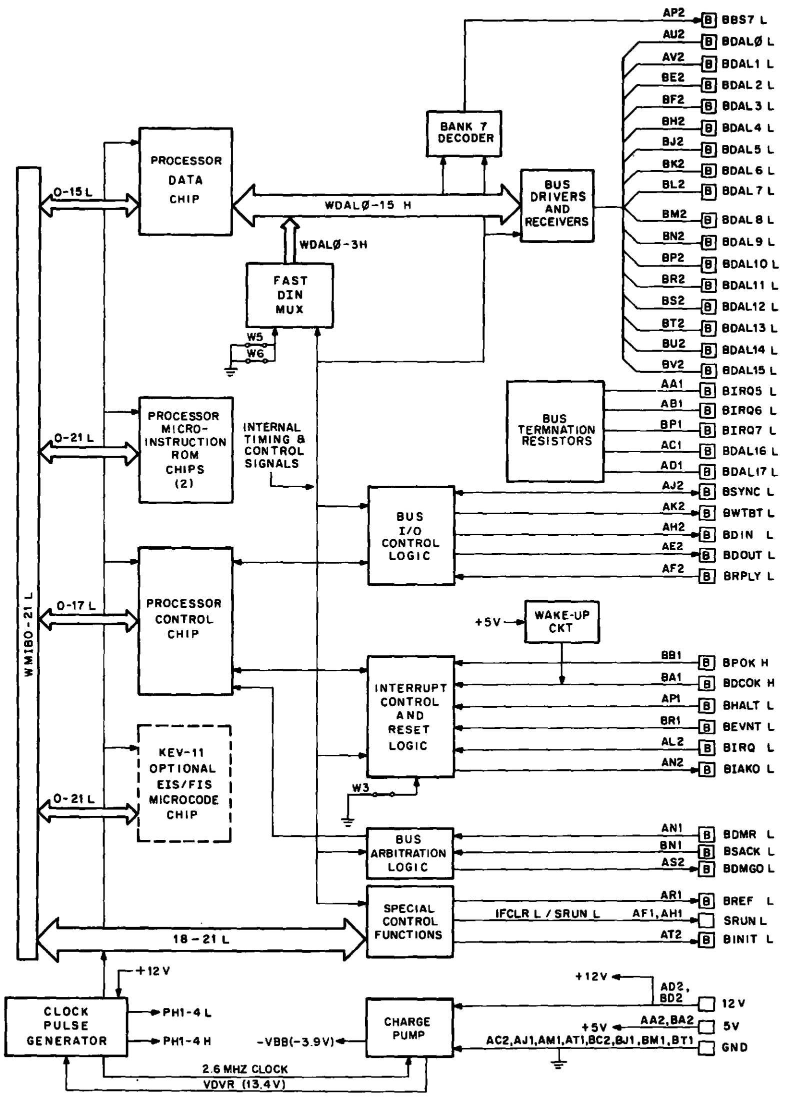
ROM Code 14 [Programmed Initialize] — A programmmed LSI-11 bus initialize operation can be performed by executing the RESET instruction. The processor responds by generating ROM Code 14 L (decoded). On the positive-going trailing edge of this signal, the Initialize flip-flop clocks to the reset (active) state, producing the active initialize signal. Approximately 10 μs later, the processor produces a TFCLR L signal, clearing the initialize signal.

During a power failure, the active DC LO L signal is distributed to the Initialize flip-flop clear input; when cleared, the flip-flop is in the active state and INIT (1) H, INIT (1) L, and BINIT L initialize signals are used to clear (or initialize) all LSI-11 system logic functions. When normal power resumes, the processor microcode terminates the initialize cycle be generating TFCLR L, presetting the Initialize flip-flop; this is the passive (noninitialize) or normal flip-flop state and all initialize signals return to their passive states.

ROM Code 15 [Fast DIN Cycle] — The processor generates this code when a fast DIN cycle is required. The fast DIN cycle allows the processor to read (input) the selected start-up mode, time-out error, and power fail signal status.

ROM Code 16 [PFCLR L] — This code clears the Power Fail flip-flop.

ROM Code 17 [EFCLR L] — This code clears the Event flip-flop (or line time clock interrupt request).

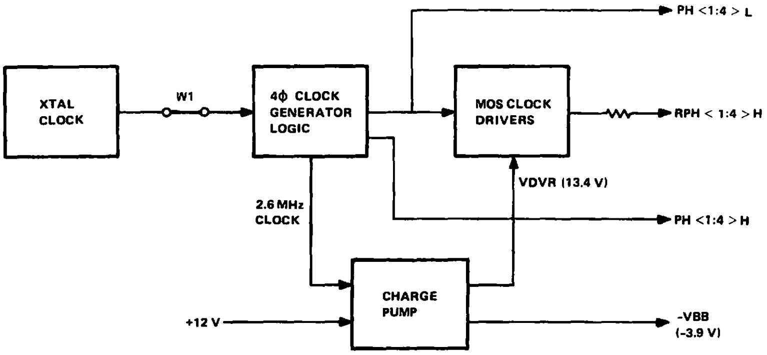


M7270 LSI-11/2 Processor Module—Basic Functions

LSI-11/2 LOGIC FUNCTIONS

The basic logic functions are shown in the accompanying figure.

Clock Pulse and Charge Pump Circuits — The clock pulse and charge pump circuits are shown in the following figure. The clock pulse generator produces 4-phase clock signals for processor timing and synchronization and a 2.6 MHz clock pulse that drives the charge pump circuit.



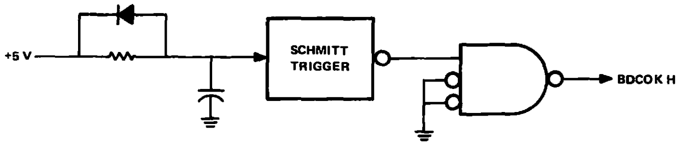
M7270 Clock Pulse and Charge Pump Circuits

The 4-phase clock generator outputs PH <1:4> L and PH <1:4> H synchronize TTL logic contained on the processor module. PH <1: 14> L signals are also applied to MOS-compatible clock drivers that produce similarly timed +12 V RPH <1:4> H signals. These signals are used for driving the 4-phase clock inputs on the processor data, control, and microinstruction ROM integrated circuits. Each clock pulse phase signal is 95 ns duration and pulses occur at 380 ns intervals.

The **charge pump** provides on-board generation of the required negative dc voltages (−5 V and −3.9 V). Input dc power for the inverter circuit is obtained directly from the +12 V input. The inverter switching rate is clocked by the clock pulse generator's DIVB (0) H 2.8 MHz output. Outputs include VDVR (+13.4 V) voltage source for the MOS clock drivers and $-V_{BB}$ (−3.9V) voltage bias for the processor data, control, and microinstruction ROM integrated circuits.

Wake-Up Circuit — The wake-up circuit causes the LSI-11 processor to self-initialize during power-up. An R-C circuit receives +5V operating power when power is turned on. When power is first applied, the low capacitor voltage causes the Schmitt trigger's output to go high and the bus driver asserts the BDCOK H signal (low). After power has been applied for approximately 1 second, the capacitor's voltage rises above the Schmitt trigger's threshold voltage, and its output goes low.

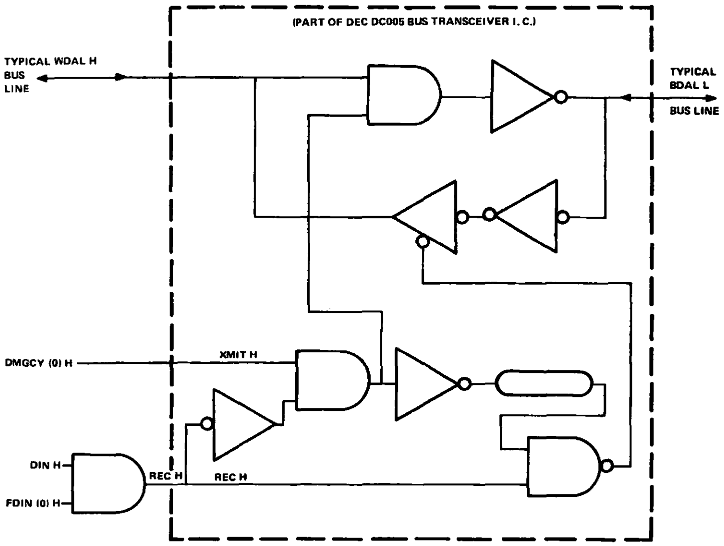
The low voltage turns off the bus driver, enabling BDCOK H to become asserted. The processor then starts its initialization sequence if no other device is asserting BDCOK H. Proper initialization requires that +12 V operating power be applied within 50 ms of +5 V operating power.



M7270 Wake-Up Circuit

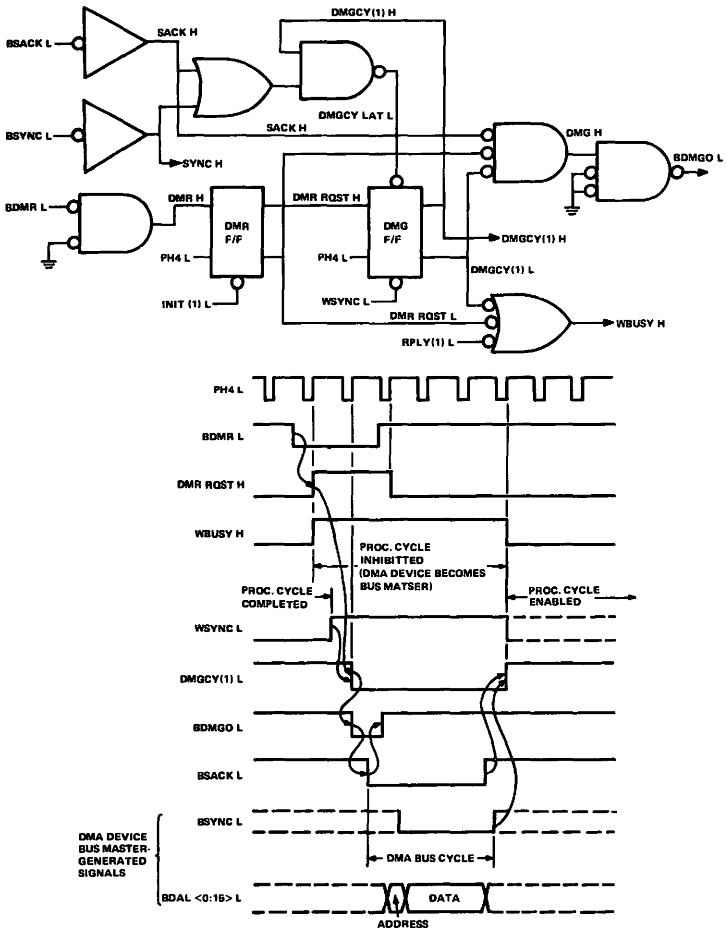
Normal operation of the wake-up circuit depends on the rise time of the +5 V power supply being faster than 50 ms. The +12 V power supply also must attain its specified operating voltage in the same 50 ms. The wake-up circuit does not provide power failure detection nor power-down sequencing. These functions, if required, must be generated externally.

BDAL Bus Driver Enable Logic — The logic circuit in the accompanying figure is used in the LSI-11/2. The four bus driver portions in each of four DEC DC005 bus transceiver integrated circuits are enabled whenever DMGCY(0) H is high (DMG cycle not in progress) and DIN H and FDIN(0) H are passive. The drivers are disabled whenever a DMG cycle is in progress, or when the processor is reading the bus [instruction fetch or data portion of DATI or DATIO(B) bus cycles]. Bus receivers are enabled only when FDIN(0) H and DIN H are both true (high).



M7270 BDAL Bus Driver Enable Logic

DMA Arbitration Logic — The DMA arbitration logic circuit used on the M7270 processor is shown in the accompanying figure. Logic functions are synchronized by the trailing edge of the PH4 L clock signal. A typical DMA arbitration (DMA request/grant) sequence is illustrated there.



M7270 DMA Arbitration Logic and Sequence

CONFIGURATION

Every LSI-11/2 processor module is factory configured to perform specific functions. For many applications the module can be used as received. Wirewrap posts are provided on each module for configuring jumper selected functions. The factory-configured functions selected are listed in the accompanying table. Processor functions may be altered by installing or removing jumpers as the following paragraphs mention.

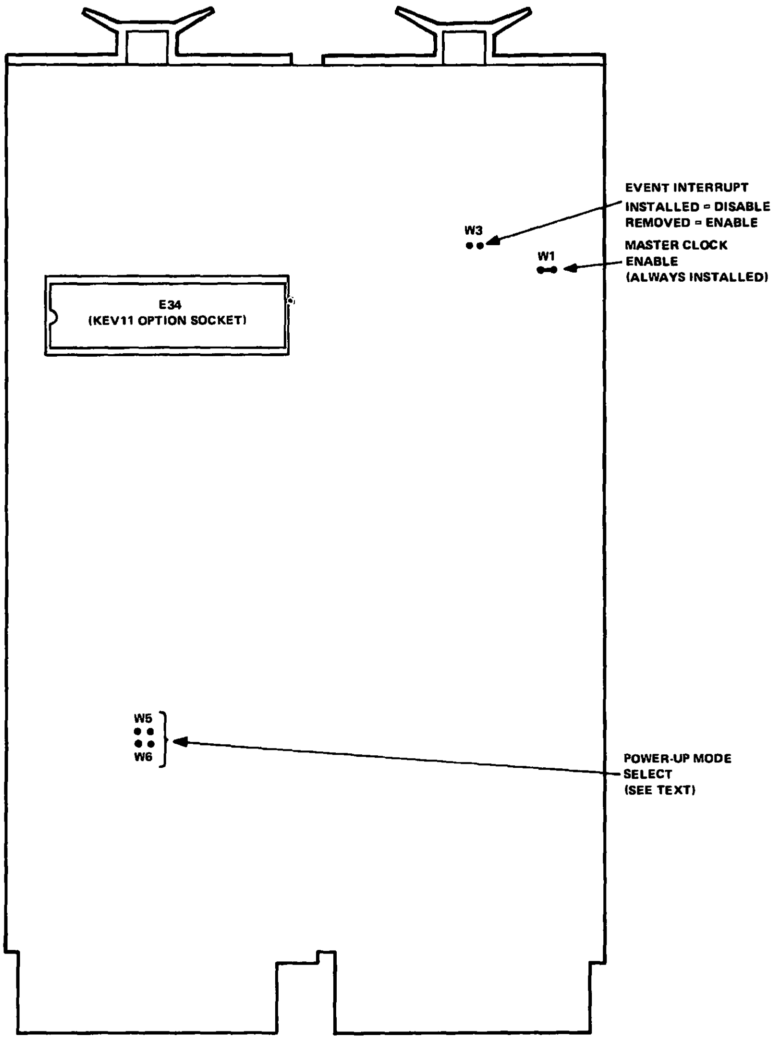
NOTES:

1. Do not change W1 on the LSI-11/2 M7270 module. It is always installed.
2. M7270 modules do not include jumpers W2, W4, and W7 through W11.

Processor module etch revisions can be determined by examining the printed circuit board part number on Side 2 (solder side) of the processor module, and as shown in the figure.

LSI-11/2 M7270 Processor Module Factory Installed Jumpers

Jumper	**Status**	**Function**
W1	I	Master clock enable (always installed—do not remove)
W2	N/A	
W3	R	Event line (LTC) interrupt enabled
W4	N/A	
W5 W6	R R	Power-up mode 0 selected
W7	N/A	
W8 W9	N/A	
W10	N/A	
W11	N/A	



M7270 Processor Module Jumper Locations

Power-Up Mode Selection — Four power-up modes are available for user selection. These are selected (or changed) by wirewrap jumpers W5 and W6 on the processor module. Note that the jumpers affect only the power-up mode (after BDCOK H and BPOK H have been asserted); they do not affect the power-down sequence.

The state of the BHALT L signal is significant during the power-up sequence. When this signal is asserted, it invokes the processor's ODT console microcode after the power-up sequence. The console device must be properly installed for correct use of the BHALT L signal.

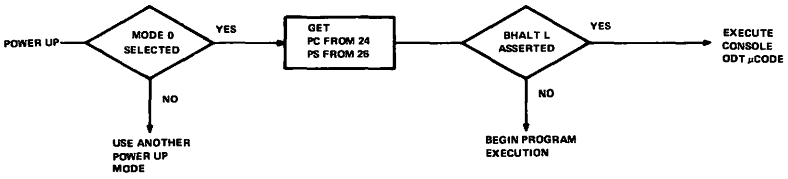
Power-up modes are listed below. Detailed descriptions of each mode are provided in the paragraphs that follow.

	Jumpers*		
Mode	**W6**	**W5**	**Mode Selected**
0	R	R	PC at 24 and PS at 26, or halt mode
1	R	I	ODT microde
2	I	R	PC at 173000 for user boot-strap
3	I	I	Special proc-essor micro-code (not im-plemented)

*R = Jumper Removed; I = Jumper Installed.

Power-Up Mode 0
This option places the processor in a microcode sequence that fetches the contents of memory locations 24 and 26 and loads their contents into the PC (R7) and the PS, respectively. A microcode service translation at this point interrogates the state of the BHALT L signal. Depending on the state of this signal, the processor either enters ODT microcode (BHALT L asserted low) or begins program execution with the current contents of R7 as the starting address (BHALT L not asserted).

Note that the T-bit (PS bit 4) is loaded with the contents of PS bit 4 in location 26. This mode should be used only with nonvolatile memory (or volatile memory with battery backup) for locations 24 and 26, or with BHALT L asserted. This power-up sequence is shown in the accompanying figure.


POWER UP
MODE 0 SELECTED
YES
NO
USE ANOTHER POWER UP MODE
GET PC FROM 24 PS FROM 26
BHALT L ASSERTED
YES
NO
EXECUTE CONSOLE ODT μCODE
BEGIN PROGRAM EXECUTION

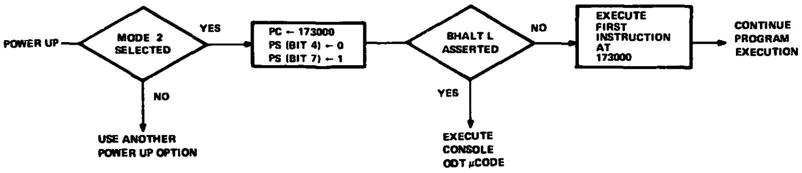
Mode 0 Power-Up Sequence

Power-Up Mode 1
This mode immediately places the processor in the console microcode regardless of the state of the BHALT L signal. This mode assumes a console interface device at bus address 177560.

Power-Up Mode 2
This mode places the processor in a microcode sequence that loads a starting address of 173000 into R7 and begins program execution at this location if the BHALT L signal is not asserted.

Note that before 173000 is loaded into R7, PS bit 4 (T-bit) is cleared and bit 7 (interrupt disable) is set. The user's program must set these bits, as desired, and set up a valid stack pointer (R6). This option should be used with nonvolatile memory (ROM, PROM, or core) at address 173000. A time-out trap through location 4 will occur if no device exists at location 173000. This mode is particularly useful when a bootstrap option is present in the system.

If BHALT L is asserted, the processor will not execute the instruction at location 173000 and will immediately execute the console microcode. This power-up mode sequence is in the accompanying figure.


POWER UP
MODE 2 SELECTED
YES
NO
USE ANOTHER POWER UP OPTION
PC ← 173000
PS (BIT 4) ← 0
PS (BIT 7) ← 1
BHALT L ASSERTED
NO
YES
EXECUTE FIRST INSTRUCTION AT 173000
CONTINUE PROGRAM EXECUTION
EXECUTE CONSOLE ODT μCODE

Mode 2 Power-Up Sequence

Power-Up Mode 3
This microcode sequence allows access to future microcode expansion in the fourth microm page (microlocations 3000 to 3777). After BDCOK H and BPOK H are asserted and the internal flags are cleared,

a microjump is made to microlocation 3002. If this option is selected and no microm responds to the fourth page microaddress, a microtrap will occur through microlocation 0 which will, in turn, cause a reserved user instruction trap through location 10.

Note that the state of BHALT L is not checked before control is transferred to the fourth microm page.

LTC Interrupt

Line time clock (LTC) or external event (EVNT) interrupts are enabled when jumper W3 is removed and the processor is running. The jumper can be inserted to disable this feature. The LTC interrupt is initiated by an external device when it asserts the BEVNT L signal. This is the highest priority external interrupt request; processor interrupts have higher priorities. If external interrupts are enabled (PS bit 7 = 0), the processor PC (R7) and PS word are pushed onto the processor stack. The LTC (or external event device) service routine is entered by vector address 100; the usual interrupt vector address input operation by the processor by the processor is not required since vector 100 is generated by the processor.

The first instruction of the service routine typically will be fetched within 16 μs from the time BEVNT L is asserted; however, if optional EIS/FIS instructions are being executed, this time could extend to 44.1 μs maximum. This time could also be extended by processor trap execution (memory refresh, T-bit, power fail, etc.), or by asserting the BHALT L signal.

PROCESSOR OPERATING CHARACTERS

Operational characteristics include: response to interrupts and traps, runaround halt modes, initialization and power fail. ODT is discussed later in this chapter.

Interrupt and Trap Priority

Interrupts and traps are similar in their operation. Interrupts are service requests from devices external to the processor; traps are interrupts that are generated within the processor. Their main operational difference, however, is that external interrupts can be recognized only when PS priority (bit 7) is zero; traps can be executed at any time, regardless of the PS priority bit status.

Traps, including EMT, BPT, IOT, and TRAP instructions, and hardware-generated Trace Trap, Bus Error, Power Fail, etc., are described in Chapter 11. The LTC (external event) interrupt has the highest priority of all external interrupts, when PS priority bit 7 = 0. This interrupt always uses vector address 100. It loads a new PC from location 100 and a new PS from location 102. All other external interrupts are requested by a device asserting the BIRQ signal. If PS bit 7 = 0, the request is acknowledged and the processor inputs a user-assigned vector address for the device service routine PC (starting address) and PS. For example, when the requesting device is the console device, vectors 60 (console input) or 64 (console output) are used. These vectors are reserved for the console device by most DIGITAL software systems.

Halt Mode

The LSI-11 microcomputer can operate in either a Run or Halt mode. When in the Halt mode, normal program execution is not performed and the processor executes ODT console microcode. However, the processor will execute memory refresh in a normal manner and arbitrate DMA requests; all external interrupts are ignored.

The Halt mode can be entered in one of six ways:

1. When the BHALT L signal is asserted
2. When a HALT instruction has been executed
3. By power-up sequence
4. When a double bus error has occurred (a bus error trap with SP (R6) pointing to nonexistent memory)
5. No Reply received from a device (bus time-out error) when the processor attempts to input a vector during an interrupt transaction
6. A bus error (time-out) occurs when the processor refreshes one of 64 memory rows

The LSI-11 microcomputer does not use conventional control panel lights and switches. Instead, the ODT console microcode routine provides all control panel features on a peripheral device that is interfaced at bus address 177560 and can interpret ASCII characters. The peripheral device used with the ODT console microcode is called the console device, which can be any device capable of interpreting ASCII characters. The prompt character sequence and detailed use of console ODT commands is explained later.

Initialization and Power Fail

Initialization occurs during a power-up or power-fail sequence, or when a RESET instruction is executed. The processor responds to these conditions by asserting the BINIT L bus signal. BINIT L can be used to clear or initialize all device registers on the bus.

During the power-up sequence, the processor asserts BINIT L in response to a passive (low) power supply-generated BDCOK H signal.

When BDCOK H goes active (high), the processor terminates BINIT L and the jumper-selected power-up sequence is executed. Similarly, if power fails, the power supply-generated BPOK H signal goes passive (low) and causes the processor to push the PC and PS onto the stack and enter a power-fail routine via vector location 24. The processor will execute a user power-fail routine until either BDCOK H goes passive (low), indicating that dc operating power may not sustain processor operation, or BPOK H returns to the active state. BINIT L will go active if BDCOK H goes passive.

Note that if a HALT instruction is executed after entering the power-fail routine, the ODT microcode will not be executed until BPOK H is reasserted. If BPOK H goes passive while the processor is in the Halt mode, the power-fail routine will not be executed.

Halt Mode

Console ODT commands are executed by the LSI-11 processor only when the processor is in the Halt mode. When in this mode, the processor responds to commands and information entered via the console terminal, and all processor response is controlled by the processor microcode.

NOTE

For console ODT communication, the serial line unit must be configured for console bus addresses 177560 through 177566. These addresses are included in the LSI-11 processor microcode and cannot be changed. If no device responds to the above addresses, bus timeout errors will occur and the processor will go into an infinite microcode loop. The only way to get out of this loop is to initialize the system (momentarily assert the BDCOK H signal low, or cycle the power off and then on).

The Halt mode is entered in one of the following ways:

- executing a HALT instruction
- pressing the BREAK key on the console terminal (this feature can be disabled by removing a jumper on the console device serial line unit)
- during power-up (power up Mode 1 configured on the processor module)
- the BHALT L bus signal is asserted

- a double Bus Error [Bus Error trap with SP (R6) pointing to nonexistent memory]
- a Bus Error (timeout) during memory refresh
- a Bus Error (timeout) when the processor is attempting to input a vector from an interrupting device

Upon entering the Halt mode, the processor outputs the following ASCII nonprinting and printing characters to the console terminal.

```
<CR><LF>
nnnnnn<CR><LF>
@
```

The nnnnnn is the location of the next instruction to be executed, and is always the contents of the PC (R7). The <CR> and <LF> are carriage return and line feed codes. The @ symbol is displayed as the prompt character for the operator; ODT will accept any of the commands described in this chapter at this point.

ODT COMMANDS

The following is a list of ODT commands and how they are used with the console terminal. Note that in the examples provided, characters output by the processor are shown underlined. Characters input by the operator are not underlined.

The commands described in this chapter are a subset of the ODT-11 utility program. Only the commands necessary for implementing the required console functions are retained.

Note also that all commands and characters are echoed by the processor and that illegal commands will be echoed and followed by ?, (ASCII 077) followed by <CR> (ASCII 015) followed by <LF> (ASCII 012) followed by @ (ASCII 100). If a valid command character is received when no location is open (e.g., when having just entered the halt state), the valid command character will be echoed and followed by a ? <CR> <LF> @. Opening nonexistent locations will have the same response. The console always prints six numeric characters as addresses or data; however, the user is not required to type leading zeros for either address or data. If a bus error (timeout) occurs during memory refresh while in the console ODT mode a ? <CR> <LF> @ will be typed.

1. **"/" Slash (ASCII 057)**

   This command is used to open a memory location, general-purpose register, or the processor status word.

   The / command is normally preceded by a location identifier.

Before the contents of a location are typed, the console prints a space (ASCII 40) character.

example:
@ 001000/021525

where:
@ = ODT prompt character (ASCII 100)
001000 = octal location in address space to be opened
/ = command to open and exhibit contents of location
012525 = contents of octal location 1000

NOTE

If / is used without a preceding location identifier, the address of the last opened location is used. This feature can be used to verify data just entered in a location.

2. **<CR> carriage return (ASCII 015)**
This command is used to close an open location. If the contents of the open location are to be changed, <CR> should be preceded by the new value. If no change to the location is necessary, <CR> will not alter its contents.

example:
@001000/12525 <CR><LF>

@/012525

or

example:

@001000/012525 15126421<CR><LF>

@/126421

where:
<CR> = (ASCII 015) is used to close location 1000 in both examples. Note that in the second example, the contents of location 1000 were changed and that only the last 6 digits entered were placed in location 1000.

3. **<LF> line feed (ASCII 021)**
This command is used to close an open location or GPR (general-purpose register). If entered after a location has been opened, it will close the open location or GPR and open location +2 or GPR +1. If the contents of the open location or GPR are to be modified, the new contents should precede the <LF> operator.

example:

```
@1000/012525<LF><CR>

001002/005252<CR><LF>

@
```

where:

<LF> = (ASCII 012) used to close location 1000 and open location 1002, if used on the PS, and <LF> will modify the PS if new data has been typed and close it; then, a <CR>, <LF>, @ is issued. If <LF> is used to advance beyond R7, the register name printed is meaningless, but the contents printed are those of R0.

4. **"↑" up arrow (ASCII 135)**
   The "↑" command is also used to close an open location or GPR. If entered after a location or GPR has been opened, it will close the open location or GPR and open location −2, or GPR −1. If the contents of the open location or GPR are to be modified, the new contents should precede the "↑" operator.

example:

```
@ 1000/012525↑ <CR><LF>

000776/010101 <CR><LF>

@
```

where:
"↑" = (ASCII 135) used to close location 1000 and open location 776.

If used on the PS, the ↑ will modify the PS if new data has been typed and close it; then <CR>, <LF>, @ is issued. If ↑ is used to decrement below R0, the register name printed is meaningless but the content is that of R7.

5. **"@"at sign (ASCII 100)**
   Once a location has been opened, the @ command is used to close that location and open a second location, using the contents of the first location as an indirect address to the second location. That is, the contents of the first location point to the second location to be opened. The contents of the first location can be modified before the @ command is used. This command is useful for stack operations.

example:

```
@ 1000/000200 @ <CR><LF>

000200/000137 <CR><LF>

@
```

where:
@ = (ASCII 100) used to close location 1000 and open location 200.

Note that the @ command may be used with either GPRs or memory contents.

If used on the PS, the command will modify the PS if new data is typed, and close it; however, the last GPR or memory location contents will be used as a pointer.

6. **"←" back arrow (ASCII 137)**
   This command is used once a location has been opened. ODT interprets the contents of the currently open word as an address indexed by the PC and opens the addressed location. This is useful for relative instructions where it is desired to determine the effective address.

   example:
   @ 1000/00200← <CR><LF>

   001202/002525 <CR><LF>

   @

   where:
   ← = (ASCII 137) used to close location 1000 and open location 1202 (sum of contents of location 1000 which is 200, 1000 and 2). Note that this command cannot be used if a GPR or the PS is the open location and, if attempted, the command will modify the GPR or PS if data has been typed, and close the GPR or PS; then a <CR> <LF> @ will be issued.

7. **"$" dollar sign (ASCII 044) or R (ASCII 122) internal register designator**

   Either command if followed by a register value 0-7 (ASCII 060-067) will allow that specific general-purpose register to be opened if followed by the / (ASCII 057) command.

   example:
   @$n/012345 <CR><LF>

   @

   where:
   $ = register designator. This could also be R.
   n = octal register 0-7.
   012345 = contents of GPR n.

   Note that the GPRs once opened can be closed with either the <CR>, <LF>, ↑, or @ commands. The "←" command will also close a GPR but will not perform the relative mode operation.

8. **"$S" (ASCII 123) processor status word**
   By replacing "n" in the above example with the letter S (ASCII 123) the processor status word will be opened. Again, either $ or R(ASCII 122) is a legal command.

   example:
   @$S/000200 <CR><LF>

   @

   where:
   $ = GPR or processor status word designator
   S = specifies processor status register and differentiates it from GPRS.

   000200 = eight bit contents of PS; bit 7 = 1, all other bits = 0.

   Note that the contents of the PS can be changed using the <CR> command, but bit 4 (the T bit) cannot be modified using any of the commands.
9. **"G" (ASCII 107)**
   The "G" (GO) command is used to to start execution of a program at the memory location typed immediately before the "G".

   example:
   @ 100 G or 100;G

   The LSI-11 PC(R7) will be loaded with 100, the PS is cleared, and execution will begin at that location. Immediately after echoing the "G", two null (000) characters are sent to the console terminal serial line unit (SLU) to act as fill characters in case the bus BINIT L signal clears the SLU. Before starting execution, a BUS INIT is issued for 10 μs idle time. Note that a semicolon character (ASCII 073) can be used to separate the address from the G and this is done for PDP-11 ODT compatibility. Since the console is a character-by-character processor, as soon as the "G" is typed, the command is processed and a RUBOUT cannot be issued to cancel the command. If the B HALT L line is asserted, execution does not take place and only the BUS INIT sequence is done. The machine returns to console mode and prints the PC followed by <CR> <LF> @.

NOTE

When the program execution begins, the serial line unit is still busy processing the two null characters. Thus, the program should not assume the done bit (bit 7) is set in the output status register at 177564.

10. **"P" (ASCII 120)**

The "P" (Proceed) command is used to continue or resume execution at the location pointed to by the current contents of the PC(R7).

example:
@ P or ;P

If the B HALT L line is asserted, a single instruction will be executed, and the machine will return to console mode. It will print the contents of the PC followed by a <CR>, <LF>, @. In this fashion, it is possible to single instruction step through a user program. However, since the BHALT L line has higher priority than device interrupts, device interrupts will not be recognized in the single step mode.

The semicolon is accepted for PDP-11 ODT compatibilty. If the semicolon character is received during any character sequence, the console ignores it.

11. **"M" (ASCII 115)**

The "M" (Maintenance) command is used for maintenance purposes and prints the contents of an internal CPU register. This data reflects how the machine got to the console mode.

example:
@ M 00213 <CR><LF>

@

The console prints six characters and then returns to command mode by printing <CR> <LF> @.

The last octal digit is the only number of significance and is encoded as follows. The value specifies how the machine got to the console mode.

Last Octal Digital Value	**Function**
0 or 4	Halt instruction or B Halt line
1 or 5	Bus error occurred while getting device interrupt vector. This error probably indicates that the priority chain (BIAKI/O L signal) is broken in the system and that an open slot exists between modules, or a device asserting BIRQ L did not latch its request. Modules must be inserted in a contiguous fashion according to the priority daisy chain.

2 or 6	Bus Error occurred while doing memory refresh.
3	Double Bus Error occurred (stack contains nonexistent address).
4	Reserved instruction trap occurred (nonexistent Micro-PD address occurred on internal CPU bus).
7	A combination of 1, 2, and 4, which implies that all three conditions occurred.

In the above example, the last octal digit is a "3", which indicates a Double Bus Error occurred.

The codes listed above are valid only when the console mode is entered, and the code is immediately displayed. This information is lost when a "G" command is issued; the code reflects what happened in the program since the last "G" command was issued.

12. **"RO" RUBOUT (ASCII 177)**
While RUBOUT is not truly a command, the console does support this character. When typing in either address or data, the user can type RUBOUT to erase the previously typed character and the console will respond with a "\" (Backslash—ASCII 134) for every typed RUBOUT.

example:

```
@ 000100/ 077777 123457 (RUBOUT)\6<CR><LF>
@0001000/123456
```

In the above example, the user typed a "7" while entering new data and then typed RUBOUT. The console responded with a "\" and then the user typed a "6" and <CR>. Then the user opened the same location and the new data reflects the RUBOUT. Note that if RUBOUT is issued repeatedly, only numerical characters are erased and it is not possible to terminate the present mode the console is in. If more than six RUBOUTS are consecutively typed, and then a valid location closing command is typed, the open location will be modified with all zeros.

The RUBOUT command cannot be used while entering a register number. R2 = / 012345 will not open register R4; however, the RUBOUT command will cause ODT to revert to memory mode and open location 4.

13. **"L" (ASCII 114)**
The "L" (Bootstrap Loader) command will cause the processor to self-size memory and then load a program that is in bootstrap

loader format (e.g., the Absolute Loader program) from the specified device. The device is specified by typing the address of its input control and status register (RCSR) immediately before the "L". No bus initialize (BINIT L signal) is issued.

example:
@ 177560L

First memory is sized, starting at 28K (157776), and the address is decremented by 2 until the highest read/write memory location is found. In small systems (e.g., 4K memory), a discernible pause of about 1 second will occur before type motion is observed. Memory refresh continues in a normal manner during the sizing process. Then, the device RCSR address (177560 in the above example) is placed in the last location in memory (XXX776) for Absolute Loader compatibility. The program is then loaded by setting the "GO" bit (bit 0) in the device address and reading a byte of data from the device address plus 2 (177562); this address is the device's receiver data buffer. PDP-11 bootstrap loader format requires that the first data byte read from the tape is 352_8. The Absolute Loader program tape, for example, has several inches of frames all punched with 351_8. The first byte following the 351_8 bytes contains the low byte of the starting address minus 1. (For Absolute Loader, this byte is 075_8.) All bytes which follow are data bytes. Loading continues until address XXX752 has been loaded. The data at that location is then treated as the low byte of a new load address. Loading continues until byte location XXX774 has been loaded. (Address detection is done via pointers contained in the LSI-11 processor's microcode.) The processor then loads a 1 into byte location XXX775 so that word location XXX774 contains a PDP-11 Branch instruction (000765). The processor does not modify the PSW nor issue a BINIT L signal; it starts program execution at location XXX774. The program being loaded must halt the processor, if that is desired. For example, when loading the Absolute Loader program, the processor will halt, and the console terminal will display XXX500 (the current PC contents), followed by <CR> <LF> @. When loading a program using the "L" command, the BHALT L signal line is ignored. If a timeout error occurs, such as would occur if a nonexistent device was entered by the user preceding the "L" command, the console will terminate the load and print ? <CR> <LF> @. Any device CSR address may be used that references an actual address configured on the reader device's bus interface controller module. For example, the console device address (RCSR = 177560) can be configured on a serial line unit which interfaces with an LT33 Teletype low-speed reader.

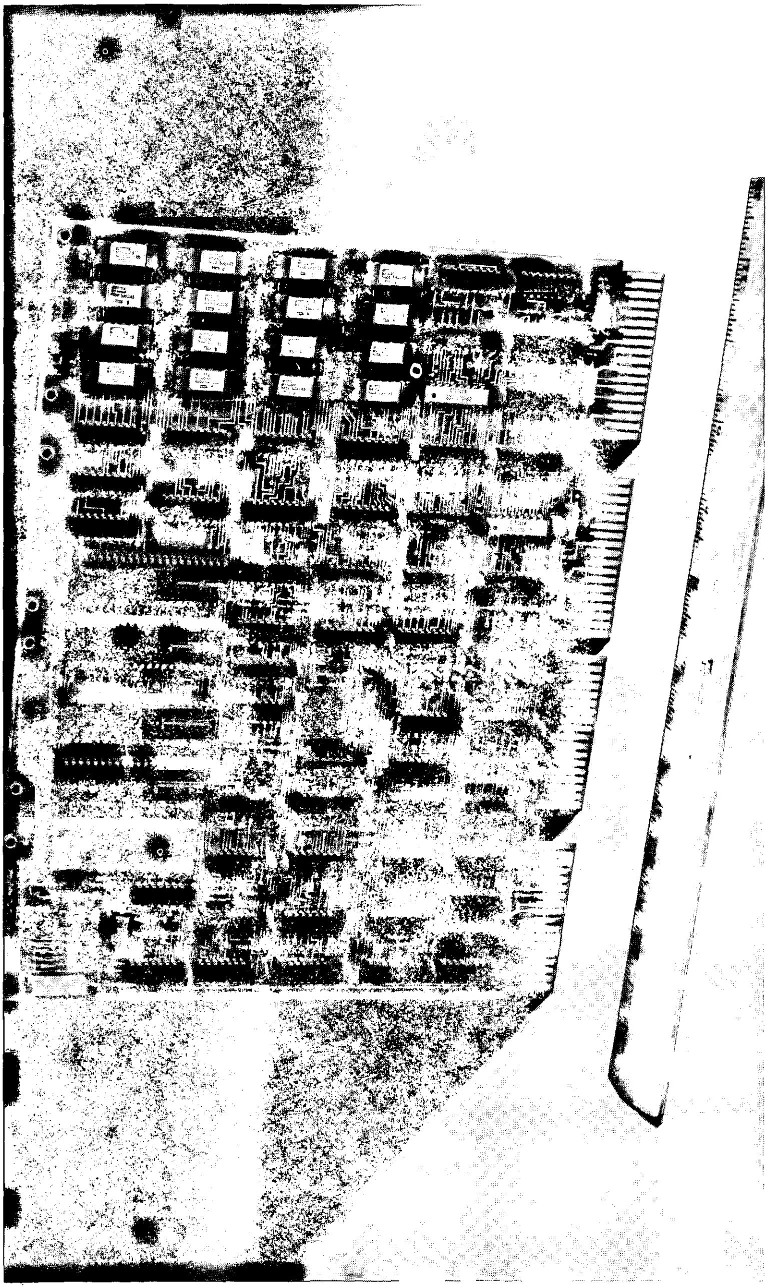
NOTE

If no address is entered for the reader device, address 0 will be used, and the system will likely "hang." The console ODT mode can be restored by momentarily asserting the BDCOK H signal low, or by cycling the power off and then on; BHALT L will have no effect on the hung condition.

14. **"CONTROL SHIFT S" (ASCII 23)**

    This command is used for manufacturing test purposes and is not a normal user command. It is briefly described here to explain the machine's response in case a user accidentally types this character. If this character is typed, ODT expects two more characters, where the first character is treated as the high byte of an address, and the second character as the low byte of an address. It uses these two characters as a 16-bit binary address, and starting at that address, dumps five locations (or ten bytes) in binary format to the serial line.

    It is recommended that if this mode is inadvertently entered, two characters such as a Null(0) and @ (ASCII 100) be typed to specify an address in order to terminate this mode. Once complete, ODT will issue a <CR> <LF> @.



CHAPTER 5
LSI-11 PROCESSOR

GENERAL

The LSI-11 is a 16-bit microcomputer with the speed and instruction set of a minicomputer. Due to its size and unique capabilities, it can fit into almost any instrumentation, data processing, or controller configuration.

A complete and powerful microcomputer system can be configured using the LSI-11, appropriate memory, I/O devices, and interconnection hardware. Communication between the system components is provided by the LSI-11 BUS.

The LSI-11 bus controls the time allocation of the LSI-11 bus for peripherals and performs arithmetic and logic operations and instruction decoding. It contains multiple high-speed, general-purpose registers which can be used as accumulators, address pointers, index registers, and for other specialized functions. The processor does both single- and double-operand addressing and handles both 16-bit word and 8-bit byte data. The bus permits DMA data transfers directly between I/O and memory without disturbing the processor registers.

FEATURES

- Extended Instruction Set (EIS) available as an option.
- Floating Point Instruction Set (FIS) available as an option.
- Writeable Control Store (WCS) available as an option.
- ODT console emulator for ease of program debugging.
- Direct addressing of 32K 16-bit words or 64K 8-bit bytes (K = 1024).
- Over 400 instructions for powerful and convenient programming.
- 16-bit word or 8-bit byte addressable locations.
- Eight internal general-purpose registers for use as accumulators and for operand addressing.
- Stack processing for easy handling of structured data, subroutines, and interrupts.
- Efficient processing of 8-bit characters without the need to rotate, swap, or mask.
- LSI-11 bus structure that provides position-dependent priority as peripheral device interfaces are connected to the I/O bus.
- Asychronous bus operation allows processor and system components (memory and peripherals) to run at their highest possible speed.

- Direct memory access (DMA) allows peripherals to access memory without interrupting processor operation.
- Fast interrrupt response without device polling.
- Power fail and automatic restart hardware detect and protect against ac power fluctuations.
- Modular component design allows systems to be easily configured and upgraded.

SPECIFICATIONS

Identification	M7246
Size	Quad
Dimensions	26.6 cm × 22.8 cm (10.5 in. × 8.9 in.)
Power Requirements	+5V ± 5%, 1.8 A +12 V ± 5%, 0.8 A
Bus Loads	ac 2.4 unit loads dc 1 unit loads
Instruction Timing	(See appendix A)
Interrupt Latency	35.05 Microsections l20% (worst case if KEV11 option not present) 44.1 microseconds ±20% (worst case if KEV11 option is present)
DMA Latency	6.45 microseconds ±20% (worst case)
ENVIRONMENTAL	
Operating Temperature	5° C to 60° C (41° to 140° F) —Derate the maximum temperature by one degree Celcius for each 1000 feet of altitude above 8000 feet.
Relative Humidity:	10% to 90%, non-condensing
Altitude:	Up to 50,000 feet (Note temperature derating above 8000 feet.)

Airflow:	Sufficient air flow must be provided to limit the temperature rise across the module to 5°C for an inlet temperature of 60°C. For inlet air temperature below 55°C, air flow must be provided to limit temperature rise across the module to 10°C.

NOTE
These are the design limits. Lower temperature limits will serve to increase the life of the module.

ENVIRONMENTAL

Storage Temperature:	−40°C to 65°C (−40°F to 149°F)
Relative Humidity:	10% to 90%, non-condensing
Altitute:	Up to 50,000 feet

NOTE
When stored outside the operating range, the module should be allowed to stabilize in the operating range for a minimum of 5 minutes before operating.

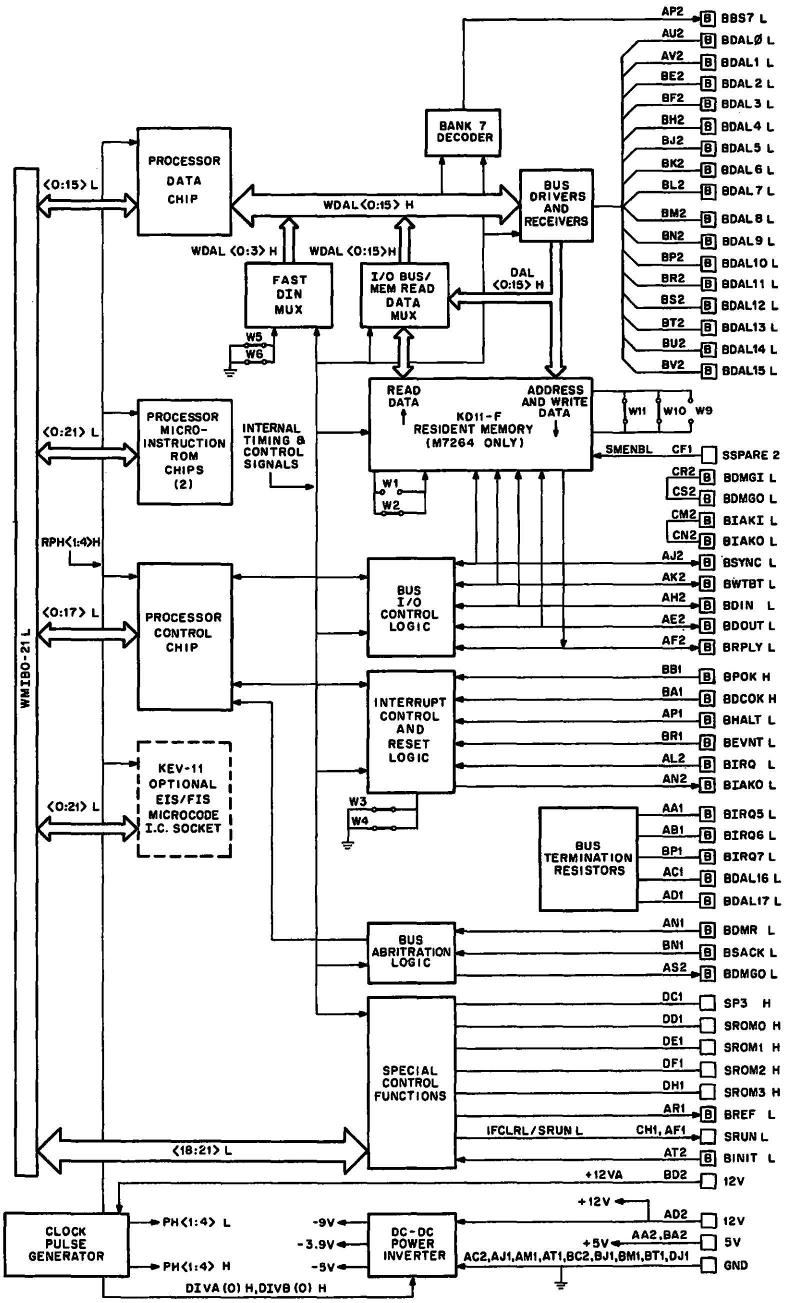
DESCRIPTION

The LSI-11 is DIGITAL's original microcomputer. There are two basic types of modules:

M7264	LSI-11 processor and 8K byte memory on a 26.6 × 22.8 cm (8.9 × 10.5 in) quad height module.
M7264-YA	LSI-11 processor only on a 26.6 × 22.8 cm (8.9 × 10.5 in) quad height module. No memory is contained on this processor module.

The following table lists the LSI-11 processor options with appropriate numbers, modules, and descriptions. A detailed description of the memory options is provided in Chapter 12.

Option	Module(s)	Description
KD11-F	M7264	LSI-11 processor and on-board 8K byte × 16-bit read/write memory
KD11-FA	M7264 M7944	LSI-11 processor and on-board 8K byte × 16-bit read/write memory plus double-height MSV11-B 8K × 16-bit read/write memory module (16K × 16-bit total memory)
KD11-FB	M7264 (2) M7944	LSI-11 processor and on-board 8K byte × 16-bit read/write memory plus two double-height MSV11-B 8K byte 16-bit read/write memory modules (24K byte × 16-bit total memory)
KD11-FC	M7264 M7955-YD	LSI-11 processor and on-board 8K byte × 16-bit read/write memory plus quad-height MSV11-CD 32K byte × 16-bit read/write memory module (40K byte × 16-bit total memory)
KD11-J	M7264-YA G653, H223	LSI-11 processor module plus quad-height MMV11-A 4K × 16-bit core memory
KD11-R	M7264-YA M7955-YD	LSI-11 processor module plus quad-height MSV11-CD 16K × 16-bit read/write memory
KD11-U	M7264-YA M8021	LSI-11 processor module plus double-height MRV11-BA ultraviolet erasable programmable read-only memory (UV PROM) 256-word read/write memory module. Memory module is supplied without UV PROM integrated circuits. Sockets are mounted on the module for user installation of PROM integrated circuits (type MRV11-BC).



M7264, M7264-YA Microcomputer Block Diagram

Microcomputer Chip Set — The main function contained on the processor module is the microprocessor chip set. This chip set includes a control chip, a data chip, and two microinstruction ROM chips (microms). In addition, an optional KEV11 microm that contains EIS/FIS microcode can be installed in a 40-pin I.C. socket contained on the module. Microprocessor chips communicate with each other over a special 22-bit microinstruction bus, WMIB <0:21> L. All address and data communication between the microprocessor chips and other processor module functional blocks is via the data chip and the 16-bit data/address lines, WDAL <0:15> H (from the data chip).

Processor module control signals interface with the microprocessor chips via the **control chip**. Eight input and five output microprocessor control signals provide this function.

The **control chip** generates a sequence of microinstruction addresses which access the microinstruction microm chips. The addressed microinstruction is then transferred to the data and control chips. Most of the microinstructions are executed by the data chip; however, various jumps, branches, and I/O operations are executed in the control chip.

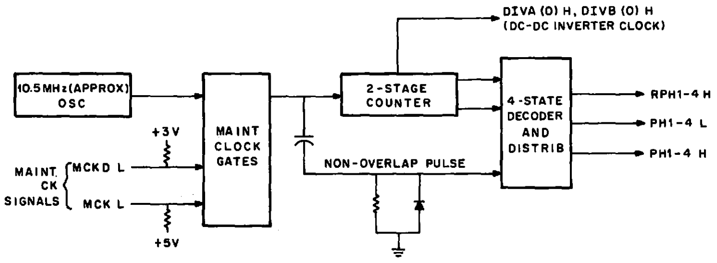
Timing and synchronization of all microprocessor chips (and all processor module functions) are controlled by four nonoverlapping clock pulses. Typical operating speed is 380 nsec (95 nsec each phase).

The **data chip** contains the data paths, logic, arithmetic logic unit (ALU), processor status bits, and registers. Registers include the eight general registers (R0-R7) and an instruction register. The user's program has access to all general registers and processor status (PS) bits. All PDP-11 instructions enter this chip via the WDAL bus. Data and addresses to and from the microprocessor are also transferred to and from the processor over this 16-bit bus.

CAUTION

Do not remove processor chips from their sockets. Improper handling will permanently damage the chips.

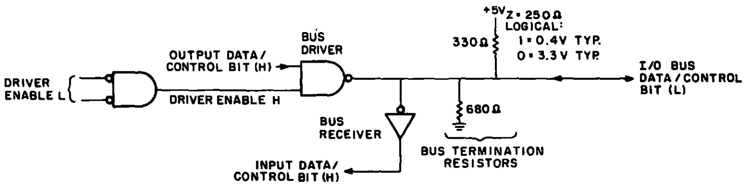
Clock Pulse Generator (M7264 and M7264-YA only) — The clock pulse generator circuit produces four nonoverlapping clock signals for processor timing and synchronization. A voltage-controlled oscillator generates a basic 10.5 MHz CK H signal.



Clock Pulse Generator

Maintenance clock gates receive and distribute the basic CK H signal to a two-stage counter and an RC filter circuit. The two-stage counter outputs are decoded by the four-state decoder, producing the basic four nonoverlapping clock phases. The pulse produced on the leading edge of each basic clock pulse inhibits the decoder for 9 ns, preventing the overlap of each phase. Each of the four phase signals (RPH1 through RPH4) are positive-going, MOS-compatible 95 ns (nominal) pulses which are bused to each of the microprocessor chips through resistors. PH1 L through PH4 L and PH1 H through PH4 H are similarly timed; however, they are TTL-compatible for distribution elsewhere on the module.

Bus Interface and Data/Address Distribution — All LSI-11 processor module communication to and from external I/O devices and memories is accomplished using the LSI-11 bus 16-bit data/address lines (BDAL <0:15> L) and bus control signals. The processor module interfaces to the bus using bus driver/receiver chips, as shown in the accompanying figure. Each bus driver/receiver chip contains four open-collector drivers and four high-impedance receivers. Each driver output is common to a receiver input. Thus, either processor output data (from the driver outputs) or input data (from the bus) can stimulate bus receiver inputs.


+5V
Z = 250Ω
LOGICAL:
1 = 0.4V TYP.
0 = 3.3V TYP.
330Ω
BUS DRIVER
OUTPUT DATA/ CONTROL BIT (H)
DRIVER ENABLE L
DRIVER ENABLE H
I/O BUS DATA/CONTROL BIT (L)
680Ω
BUS RECEIVER
BUS TERMINATION RESISTORS
INPUT DATA/ CONTROL BIT (H)

LSI-11 Bus Loading and Driver/Receiver Interface

Note that all four drivers in a chip are enabled or disabled by a pair of DRIVER ENABLE L inputs. A **high** input will inhibit all four drivers; when both enable inputs are **low**, the drivers are enabled and output data is gated onto the bus. Signals on the M7264 or M7264-YA module which control bus drivers include EDAL L, INIT (1) H, and DMGCY H. False states enable certain control signals described later.

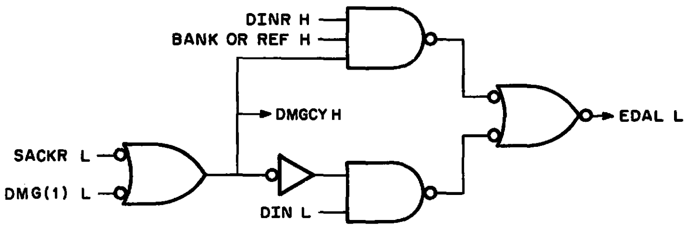
EDAL L is a control signal which enables the 16-bit data/address bus drivers on M7264 and M7264-YA modules only. When in the active state, EDAL L gates WDAL <0:15> H onto the BDAL <0:15> L bus. EDAL L is generated by the logic shown in the accompanying figure. During a processor-controlled address/data output bus cycle, or during the addressing portion of a processor-controlled input bus cycle, SACKR L and DMG(1) L are passive (high). The passive signals are gated, producing a low (passive) DMGCY H signal. This signal is inverted and gated with the passive DIN L signal, producing the active EDAL L signal. During a DMA cycle in which data in the processor module resident 4K memory is to be read by a DMA device, BANK OR REF H goes high; this signal is gated with DINR H and DMG CYCLE H to produce the active EDAL L signal.

DMGCY H and INIT (1) H are processor module logic control signals which inhibit certain bus drivers during an Initialize or DMA operation. Bus drivers are enabled when these signals are in the false (low) state.

A list of bus driver output signals and their respective enable signals is provided below.

Bus Driver	**Enable Signal(s)**
(Signal)	**(Low = Enable)**
BDAL0-15L	EDAL L (M7264 and M7264-YA processor modules only)
BSYNC L	
BBS7 L	INIT (1) H, DMGCY H
BREF L	
BIAKO L	
BWTBT L	
BRPLY L	
BDIN L	INIT (1) H
BDOUT L	
BINIT L	Always enabled
BDMGO L	

The near-end bus termination resistors are contained on the processor module. Each bus driver output is terminated by a pair of resistors, as shown in the figure, establishing the nominal 250Ω bus impedance and the 3.4 V nominal voltage level.



EDAL L Logic

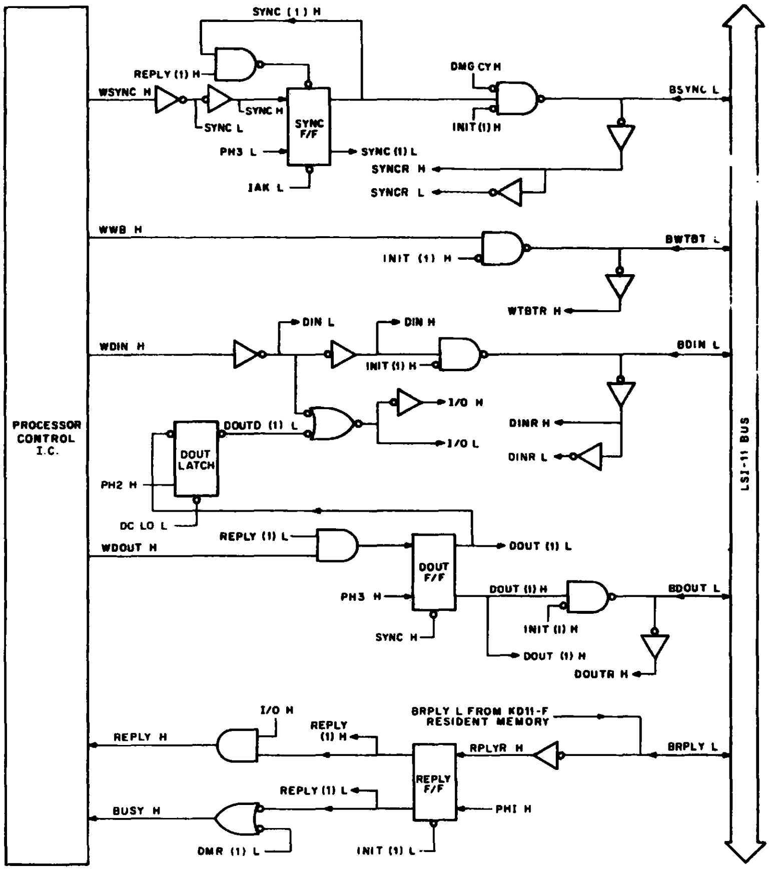
Address and data information are distributed on the processor module via the WDAL <0:15> H and DAL <0:15> H 16-bit buses. WDAL <0:15> H interface directly with the microprocessor data chip, the DEC 8641 bus drivers, and, on M7264 processor modules (only), the I/O bus/memory read data multiplexer. All processor input data from the I/O bus is via the bus receivers, the DAL <0:15> H bus, the data multiplexer, the WDAL <0:15> H bus, and the microprocessor data chip. (Resident memory data input is discussed later.)

Bus I/O Control Signal Logic — Bus I/O control signals include BSYNC L, BWTBT L, BDIN L, BDOUT L, and BRPLY L. In addition, BIAKO L can be considered a bus I/O control signal; however, since it is only used during the interrupt sequence, it is discussed later. Logic circuits which produce and/or distribute these signals are shown in the accompanying figure. Each signal is generated or received as described in the following paragraphs.

BSYNC L — The control chip initiates the BSYNC L signal sequence by raising WSYNC H during PH2. Inverters apply the high SYNC H signal to the Sync flip-flop sets, producing an active (high) SYNC (1) H input to the BSYNC L bus driver. SYNC (1) H is gated with REPLY (1) H (when active) to produce a direct preset input to the Sync flip-flop. This ensures that BSYNC L will remain active until after the bus slave device terminates its BRPLY L signal and the Reply flip-flop is reset. [REPLY (1) H is low.] The Sync flip-flop then clocks to the reset (BSYNC L passive) state on the trailing edge of PH3 L.

BWTBT L — BWTBT L is the buffered/inverted control chip WWB H output signal. This signal asserts during PH1 of the addressing portion of a bus cycle to indicate that a write (output) operation follows. It remains active during the output data transfer if a DATOB bus cycle is to be executed.

BDIN L — BDIN L is the inverted, buffered control chip WDIN H signal. This signal goes active during PH2 following an active RPLY H signal.



Bus I/O Control Signal Logic

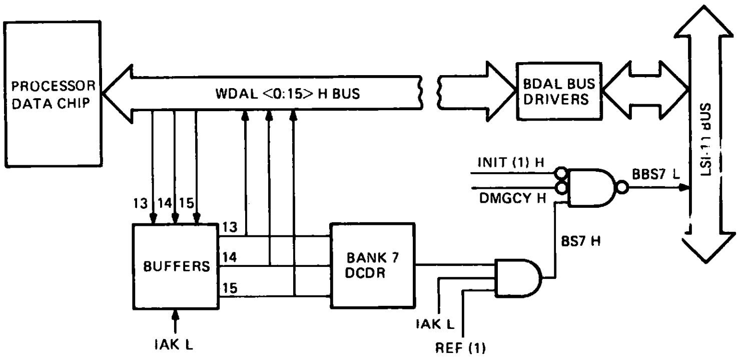
BDOUT L — The control chip initiates the BDOUT L signal sequence by raising WDOUT H during PH2. This signal is gated with the passive REPLY (1) L (high) signal to produce an active (low) D input to the DOUT flip-flop. The flip-flop sets on the leading edge of PH3 H, producing an active BDOUT L signal. It clocks to the reset state on PH3 following the REPLY (1) active (low) signal.

BRPLY L — BRPLY L is a required response from a bus slave device during input or output operations. DIN L and DOUT (1) L are ORed to produce an active I/O L signal whenever a programmed transfer occurs.

I/O L enables the time-out counter in the bus error detection portion of the interrupt logic. I/O is inverted to produce I/O H, which enables the reply gate REPLY H signal input to the control chip.

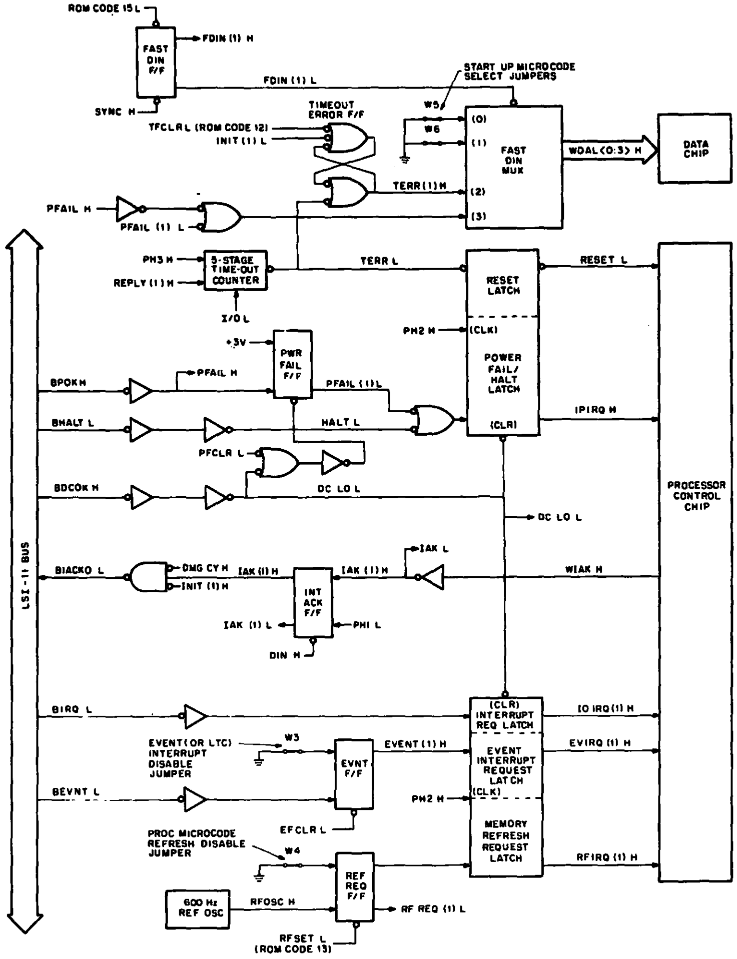
BRPLY L is received either from the LSI-11 bus or resident memory and inverted to produce a high input to the Reply flip-flop. PH1 H clocks the flip-flop to set state, producing active REPLY (1) H and REPLY (1) L signals. REPLY (1) L is ORed with DMR (1) L to produce an active BUSY H signal. The control chip responds by entering a wait state, inhibiting completion of the processor-generated bus transfer for the duration of REPLY (1) L. REPLY (1) H is gated with I/O H to produce an active REPLY H signal, informing the processor that the output data has been taken or that input data is available on the bus. REPLY H goes passive when I/O H goes passive. The bus slave device will then terminate the BRPLY L signal, indicating that it has completed its portion of the data transfer. On the next PH1 H clock pulse, the Reply flip-flop resets and REPLY (1) H and L and BUSY H go passive.

Bank 7 Decoder — The bank 7 decode circuit is shown in the accompanying figure. Buffers receive WDAL <0:15> H bits and distribute them to the bank 7 decoder and BDAL bus drivers. Bank 7 is decoded during the addressing portion of the bus cycle. If a peripheral device address is referenced, an address in bank 7 (28-32K address space) is used, and WDAL <13:15> H are all active (high). This address is decoded and BBS7 L is asserted. When active, BBS7 L enables addressing of nonmemory devices along the bus. During interrupt vector bus transactions, IAK L becomes asserted. IAK L inhibits BS7 H and BBS7 L generation, which could result in an invalid input data transfer. REF(1) L inhibits BS7 H and BBS7 L generation during memory refresh bus cycles.



Bank 7 Decoder

Interrupt Control and Reset Logic — Interrupt control and reset logic functions are shown in the accompanying figure. Reset functions include bus error and power-fail (BDCOK H negated). Interrupt functions include power-fail (impending), Halt mode (console microcode control), refresh interrupt, event (or line time clock) interrupt, and external BIRQ interrupts. The various functions are described in the following paragraphs.



Interrupt Control and Reset Logic

Power-Fail/Restart Sequence — A power-fail sequence is initiated when BPOK H goes low, clocking the Power-Fail flip-flop to the set state. PFAIL (1) L is ORed with HALT L to produce a high signal. This signal is latched during PH2 H, producing an active IPIRQ H (interrupt 1) input to the control chip. The processor then interrupts program execution. Note that the low (passive) BPOK H signal is inverted to produce an active PFAIL H input to the fast DIN multiplexer; this signal status is checked by the microcode to ensure that BPOK H is asserted.

Upon entry to this microcode routine, the processor requests a fast DIN cycle. This request is decoded as ROM CODE 15 L, presetting the fast DIN flip-flop. FDIN (0) H goes low, enabling the fast DIN multiplexer to place power-up mode option jumper data, the passive time-out error [TERR (1) H] signal, and the active PFAIL H signal on WDAL <0: 3> H. The processor receives the fast DIN information via the data chip. An active PFAIL H signal informs the processor that a power-fail condition is in progress, rather than the halt condition.

BDCOK H goes passive (low) and produces an active DC LO L signal, clearing the Power-Fail flip-flop and the power-fail/halt and reset latches and initializing the processor and all devices. The active RESET L signal then initializes the processor, causing it to abort console (halt) or power-fail microcode execution and enter a "no operation" state. The processor remains in this condition until BDCOK H returns to the active state.

Once initiated, the power-fail sequence must be completed before the power-up sequence is started, otherwise the processor will "hang."

The power-up restart condition occurs when DC LO L goes false; RESET L goes passive (high) on the next PH2 H clock pulse. The processor responds by executing a fast DIN cycle to determine the start-up microcode option jumper configuration. Once the fast DIN cycle has been completed, the processor executes the power-up option selected, and normal operation resumes when BPOK H is asserted.

Halt Mode — The Halt mode is entered by executing the HALT instruction, by a device asserting the BHALT L signal, by a double bus error condition, or by a bus error (time-out) during an interrupt. The processor halts program execution and enters microcode execution as described for a power-fail operation. However, when the processor executes the fast DIN cycle, the PFAIL H bit (WDAL3 H) is not active and console microcode (not a power-fail sequence) is executed. Negation of BHALT L will allow the processor to resume PDP-11 program execution. On the next PH2 H clock pulse, IPIRQ H goes false (low) and the processor Run mode is enabled.

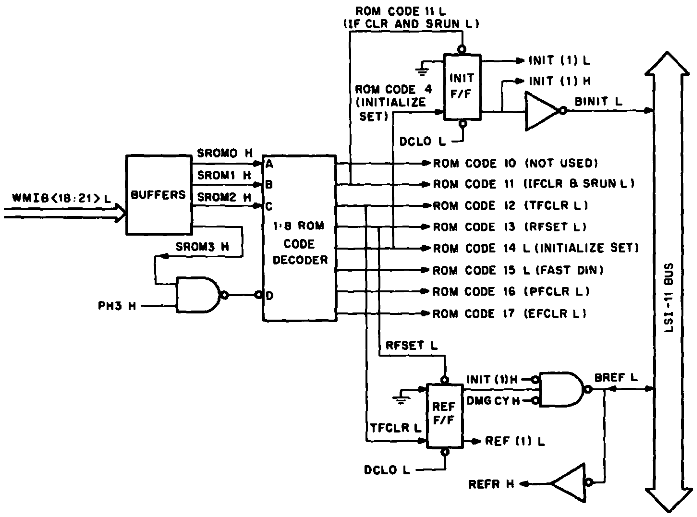
Bus Errors — A bus error results in aborting program execution and entry into a trap service routine via vector location 004. A bus error occurs when a device fails to respond to the processor DBIN L or DBOUT L signal by not returning a BRPLY L signal within 10 μs (approximately). An active I/O signal inhibits the reset input of the 5-stage time-out counter, enabling counter operation. (When not in a processor-controlled bus I/O cycle, I/O L is passive (high), clearing the counter.) The counter proceeds with counting PH3 H clock pulse signals. Normally BRPLY L would be asserted, producing an active REPLY (1) H signal which inhibits the counter; the count would remain stable until cleared by a passive I/O L signal. However, if BRPLY L is not received within 10 μs, the full count (32_{10}) is attained. This is the error condition; TERR L goes low and TERR (1) H goes high. The next PH2 H clock pulse clocks the reset latch to the reset (active) state, producing an active RESET L signal. The processor responds by executing the reset microcode. After entering the microcode, the processor executes a fast DIN cycle and determines that a time-out (bus) error TERR (1) H, rather than a power-fail condition, has occurred. It then responds by executing the bus error trap service routine. TFCLR L (ROM code 2) is generated by the processor to clear the TERR latch.

Normal I/O Interrupts — "Normal" I/O interrupts are those interrupt requests which are generated by external devices using bus interrupt request BIRQ L. The request is initiated by asserting BIRQ L. This signal is inverted to produce a high signal, which is stored in the interrupt request latch on the next PH2 H pulse. The stored request produces IOIRQ (1) H, which informs the processor of the request. If processor status word priority is 0, the processor responds by producing an active WIAK H (interrupt acknowledge) and WDIN H signals. WDIN H is buffered onto the BDIN L signal line to signal devices to stabilize their priority arbitration. WIAK H is inverted, producing IAK L, setting the Interrupt Acknowledge flip-flop on the trailing edge of PH1 L one cycle after BDIN L is asserted. The high (active) interrupt acknowledge signal is enabled onto the BIAKO L signal line by passive (low) DMGCY H and INIT (1) H signals. The highest priority device requesting interrupt service responds to the processor BDIN L and BIAK L signals by placing its vector on the BDAL bus and asserting BRPLY L, inputting its vector to the processor. Note that BSYNC L is not asserted during this operation and that no device addressing occurs. The device also clears its BIRQ L signal. The processor responds to BRPLY L by terminating BDIN L and BIAK L.

Refresh — Memory refresh is initiated by a 600 Hz refresh oscillator. This function is enabled when jumper W4 is not installed. The leading edge of RFOSC H clocks the Refresh Request flip-flop to the set state. On the next PH2 H clock pulse, the memory refresh request latch stores the request and applies an active RFIRQ H signal to the processor control chip. The processor responds by producing an active RF SET L signal and executing the refresh microcode. RF SET L sets the Refresh Request flip-flop, producing the BREF L signal and clearing the Refresh flip-flop, which terminates the request. TFCLR L resets the Refresh flip-flop when the refresh operation is completed. Note that BREF L is not asserted if DMGCY H or INIT (1) H is asserted.

Event Line Interrupt — The event line interrupt function can be used as a line time clock interrupt, or as desired by the user. This interrupt differs from the normal I/O interrupt request by being the highest priority external interrupt, and it does not input a vector in order to enter its service routine. The interrupt is initiated by the external device by asserting BEVNT L. This signal is inverted to produce a high (active) signal, which clocks the Event flip-flop to the set state. (Note that when W3 is installed, the flip-flop remains reset and the event function is disabled.) On the next PH2 H clock pulse, the event interrupt request latch stores the active EVNT (1) H signal. An active EVIRQ (1) H signal is then applied to the control chip. If processor status word priority is 0, the interrupt will be serviced. Service is gained via vector 100_8, which is dedicated to the event interrupt. Hence, bus DIN operation does not occur when obtaining the vector. The request is cleared by the microcode-generated EFCLR L signal.

Special Control Function — Special control functions include microcode-generated bus initialize and memory refresh operations and five special control signals which are internally on the processor module. Special control function logic circuits are shown in the accompanying figure. Micro-instruction bus lines WMIB <18:21> L are buffered to produce the four SROM <0:3> H signals. The actual codes for the special functions are contained on SROM <0:2> H; SROM3 H is always active when a special function is to be decoded, enabling the 1:8 ROM code decoder during PH3 H. The resulting decoded functions are described below.



Special Control Functions

ROM Code 10 — Not used.

ROM Code 11 [IFCLR and SRUN L] — This code is produced by the processor to clear the initialize flip-flop and to assert the SRUN L signal for an external RUN indicator circuit.

ROM Code 12 [TFCLR L] — This code is a trap function clear signal which clears the Refresh Request and Time-Out Error flip-flops.

ROM Code 13 [RFSET L] — This code is used to set the Refresh flip-flop. The active (high) flip-flop output is gated with passive (low) INIT (1) H and DMG (1) H signals to produce the active BREF L signal. The flip-flop normally resets by the microcode-generated TFCLR L signal after completing the refresh operation, or whenever a power failure occurs. (DC LO L goes active and clears the flip-flop.)

ROM Code 14 [Programmed Initialize] — A programmed LSI-11 bus initialize operation can be performed by executing the RESET instruction. The processor responds by generating ROM Code 14 L (decoded). On the positive-going trailing edge of this signal, the initialize flip-flop clocks to the reset (active) state, producing the active initialize signal. Approximately 10 μs later, the processor produces a TFCLR L signal, clearing the initialize signal.

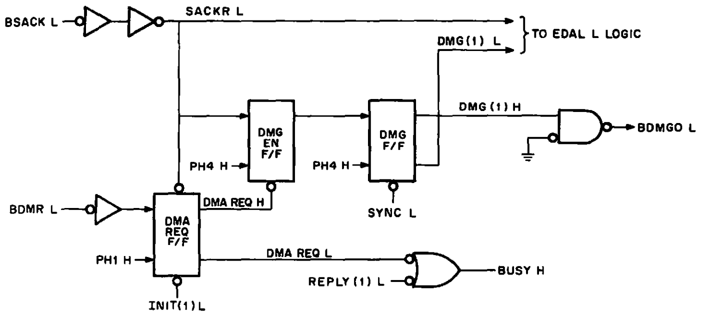
During a power failure, the active DC LO L signal is distributed to the initialize flip-flop clear input; when cleared, the flip-flop is in the active state and INIT (1) H, INIT (1) L, and BINIT L initialize signals are used to clear (or initialize) all LSI-11 system logic functions. When normal power resumes, the processor microcode terminates the initialize cycle by generating TFCLR L, presetting the Initialize flip-flop; this is the passive (noninitialize) or normal flip-flop state and all initialize signals return to their passive states.

ROM Code 15 [Fast DIN Cycle] — The processor generates this code when a fast DIN cycle is required. The fast DIN cycle allows the processor to read (input) the selected start-up mode, time-out error, and power fail signal status.

ROM Code 16 [PFCLR L] — This code clears the Power Fail flip-flop.

ROM Code 17 [EFCLR L] — This code clears the Event flip-flop (or line time clock interrupt request).

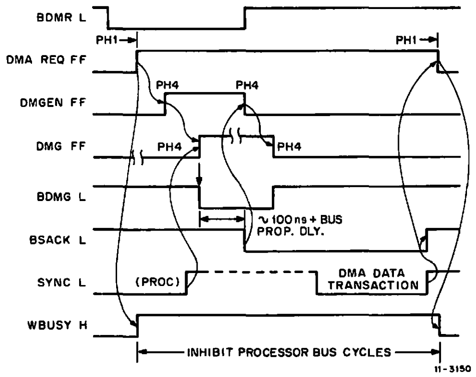
M7264 and M7264-YA Bus Arbitration Logic — Bus arbitration logic enables the LSI-11 bus to be used by DMA devices or the processor. The device (or processor) controlling the bus is called the bus master. When no DMA requests are pending, the processor is bus master and all data transfers are programmed. When a DMA device is bus master, processor operation is suspended until the DMA operation is finished.



M7264 Bus Arbitration Logic

Prior to a DMA request, the DMA Request flip-flop is reset; the DMA REQ H signal is passive (low), clearing the DMG Enable flip-flop. A device initiates a DMA request by asserting BDMR L. The request is inverted to produce a high signal, which is clocked into the DMA Request flip-flop on the next PH1 H clock pulse, producing active DMA REQ H and L signals. DMA REQ L is ORed with REPLY (1) L, producing BUSY H and causing the processor to "wait" after completing its present bus cycle. On the leading edge of PH4 H, the stored DMA request sets the DMG Enable flip-flop. The processor is finished with its present bus cycle and releases the bus when SYNC L goes passive (high).

On the first PH4 H clock pulse following the passive state of SYNC L, the DMG flip-flop clocks to the set state and DMG (1) H and DMG (1) L go to their active states. DMG (1) H produces the active BDMG grant (BDMGO L) signal. DMG (1) L enables EDAL L signal generation when the DMA operation involves M7264 resident memory. The DMA device responds to the BDMG signal by negating BDMR L and asserting BSACK L, enabling EDAL L signal generation and keeping the DMA Request flip-flop in the set state. On the first PH4 H clock phase following the active state of BSACK L, the DMG Enable flip-flop clocks to the reset state and DMG EN H goes low. The following PH4 H clock pulse clocks the DMG flip-flop to the reset state and BDMGO L goes passive (high), terminating the DMA request/grant sequence. BSACK L remains asserted for the duration of the DMA operation, preventing new DMA requests from being granted.



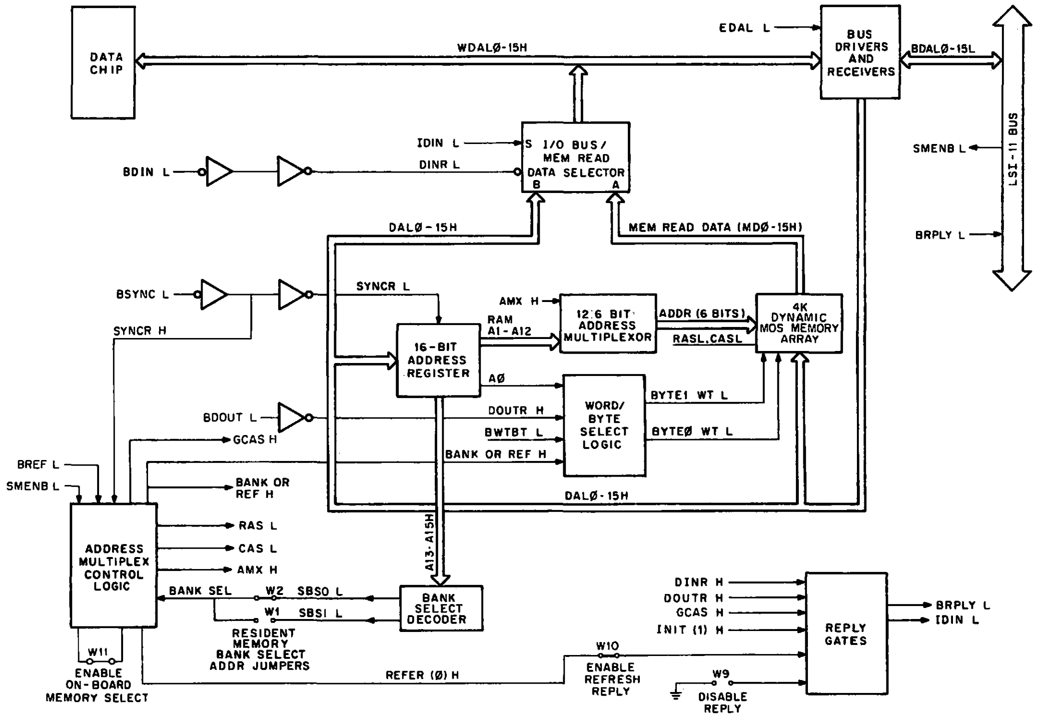
M7264 DMA Grant Sequence

The DMA device releases the bus by terminating BSACK L. The following PH1 H clock pulse clocks the DMA request flip-flop to the passive state. BUSY H then goes passive, enabling a processor-initiated bus cycle. Once the processor-initiated cycle is entered, SYNC L inhibits (clears) the DMG flip-flop for the duration of the processor's present bus cycle.

M7264 and M7264-YA DC-DC Power Inverter — The dc-to-dc power inverter circuit provides on-board generation of required negative dc voltages (−5 V and −3.9 V). Input dc power for the inverter circuit is obtained directly from the +12 V input. The inverter switching rate is clocked by the clock pulse generator's DIVB (0) H 2.8 MHz output.

M7264 Resident Memory — The 8K byte by 16-bit dynamic MOS read/write memory is included on the M7264 (KD11-F) processor module only. Resident memory can reside in either the first or second 8K byte address bank. One of two jumpers can be installed on the module to select the desired bank (bank 0 or 1).

The basic functions involving the resident memory are shown in the accompanying figure. Resident memory comprises sixteen 8K byte by 1-bit memory chips, addressing, and control logic. The memory chips, which are 16-pin devices, require an address multiplexer to address the chips with two 6-bit bytes. The complete addressing, write, and read operations are described in the next paragraph.



M7264 Resident Memory

Addressing is initiated by a master device—either the LSI-11 processor or a DMA device—by placing the 16-bit address on BDAL <0:15> L and asserting BSYNC L, latching the address in the 16-bit address register. Note that the resident memory address will appear on the BDAL bus even when the processor is bus master; the resident memory functions exactly as a memory located elsewhere along the LSI-11 bus. Address bits are routed via BDAL bus receivers onto the processor module DAL <0:15> H bus to the address register input. Stored address bits A <13:15> H are then decoded by the bank select decoder. SBS0 L (bank 0) and SBS1 L (bank 1) will go active (low) only when their respective bank addresses are decoded. W1 or W2 and W11 then applies the selected address to the address multiplex control logic, enabling the resident memory response. Address multiplex logic immediately generates an active row address strobe (RAS), which remains active for the duration of the BSYNC L signal. Address multiplex control (AMX) is initially high, multiplexing the stored row address (bits A <7:12> H) through the 12:6 bit address multiplexer and into all memory chips. After 150 ns, address multiplex control logic generates an active column address strobe (CAS) and a low AMX signal. The multiplexer output bits are then strobed into all memory chips, completing the addressing portion of memory operation.

Resident memory bank selection can be accomplished by an external signal. W11 is removed to disable the on-board memory bank selection. SMENB L is then asserted low by the external circuit to select the resident memory.

When in memory read operation, each of the 16 memory chips places an addressed bit on the memory read data bus. This data is multiplexed via port A of the I/O bus/memory read data selector only when in a resident memory read (or refresh) operation; the select input of the data selector is asserted low for this data selection. The read data is then placed on WDAL <0:15> H, where it can be read by the microprocessor data chip or gated onto the BDAL bus via bus drivers for input to a DMA device.

When in a memory write operation, the addressing portion of the operation is similar to the read cycle addressing, except BWTBT L may be asserted by the master device to indicate that a write operation is to follow. After the addressing portion of the cycle has been completed, BWTBT L either goes passive (high) if a DATO (word) write cycle is to be performed, or remains asserted (BWTBT L remains low) if a DATOB (byte) write cycle is to be performed. Word 1 byte select logic responds to the DATO cycle by asserting both BYTE 1 WT L and BYTE 0 WT L for the duration of the cycle, enabling DAL <0:15> H data bits into the addressed location in all memory chips. However, when in a

DATOB cycle, only one active signal is produced, depending upon the state of the stored byte pointer (address bit A0). If A0 is low (even byte), only BYTE 0 WT L goes active, enabling only DAL <0:7> H bits to be written into the addressed location in the appropriate eight memory chips. Similarly, if A1 is high (odd byte), only BYTE 1 WT L goes active, enabling only DAL <8:15> H bits to be written into the addressed location in the appropriate eight memory chips.

Resident memory, as well as any LSI-11 bus device, must respond to any data transaction by generating an active BRPLY L signal. Reply gates provide this function. Approximately 150 ns after CAS L goes true (as previously described), the reply gates are enabled; the gates will respond to either an active BDIN L or BDOUT L signal by asserting BRPLY L. Reply gates are inhibited during an initialize operation.

Resident memory requires a refresh operation once every 1.6 msec. This operation is entirely under the control of either processor microcode or an external DMA device, as selected by the user. Resident memory responds to BREF L, generated by the refresh-controlling device, by simulating a "bank selected" operation. (All memory banks can be simultaneously refreshed.) Refresh is then accomplished by executing 64 successive BSYNC L/BDIN L operations while incrementing BDAL <1:6> L by one location on each bus transaction. Refresh is simply a series of forced memory read operations where only the row addresses are significant. Each of the 64 rows are simultaneously refreshed in this manner.

CONFIGURATION

Configuring Processor Module Jumpers

LSI-11 processor module is factory configured for specific functions. In many applications the processor module can be used as received. Wirewrapped posts are provided on each module for configuring jumper-selected functions. The factory-configured functions selected are listed in the accompanying table. Install or remove jumpers to alter processor functions as directed.

NOTES

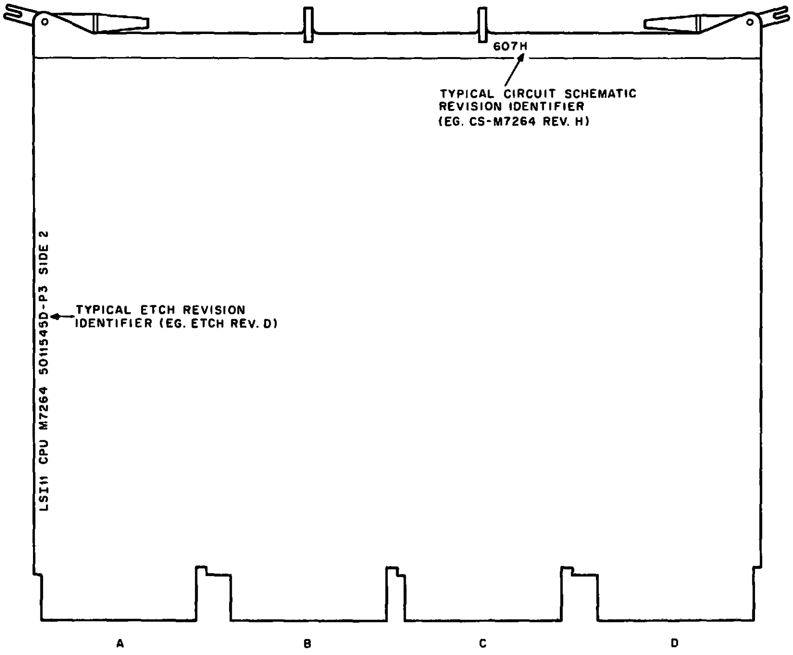
1. Do not change the following factory-configured jumpers:
   M7264 and M7264-YA modules (etch revision E and later):
   W7 and W8
2. M7264 and M7264-YA etch revision C and D modules do not include jumpers W7 through W11.

Processor module etch revisions can be determined by examining the printed circuit board part number on Side 2 (solder side) of the processor module. This number is located on M7264 and M7264-YA modules as shown in the accompanying figure. Jumpers for M7264 and M7264-YA LSI-11 processor modules are located as illustrated in the figure.

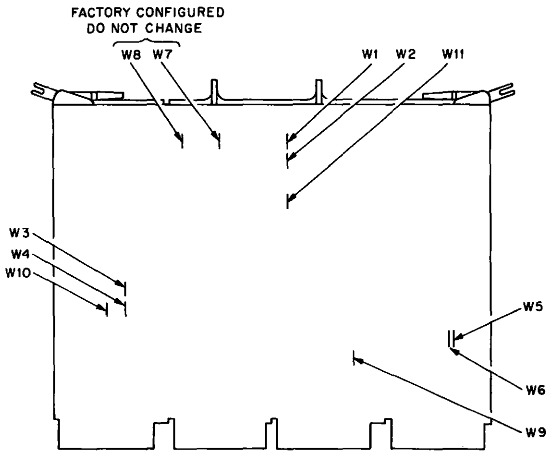
LSI-11 Processor Module Factory-Installed Jumpers

	M7264		**M7264-YA**	
Jumper	**Status**	**Function**	**Status**	**Function**
W1	R	Resident memory bank 1 not selected	R	Resident memory bank 1 not selected
W2	I	Resident memory bank 0 selected	R	Resident memory bank 0 not selected
W3	R	Event line (LTC) interrupt enabled	R	Event line (LTC) interrupt enabled
W4	R	Processor-controlled memory refresh enabled	I	Processor-controlled memory refresh disabled
W5	R	Power-up mode	R	Power-up mode
W6	R	0 selected	R	0 selected
W7	—	Factory-configured bias voltage (do not change)	—	Factory-configured bias voltage (do not change)
W8	—		—	
W9	R	Enable reply from resident memory	I	Disable reply from resident memory
W10	R	Enable reply from resident memory during refresh	R	N/A
W11	I	Enable on-board memory select	R	Disable on-board memory select

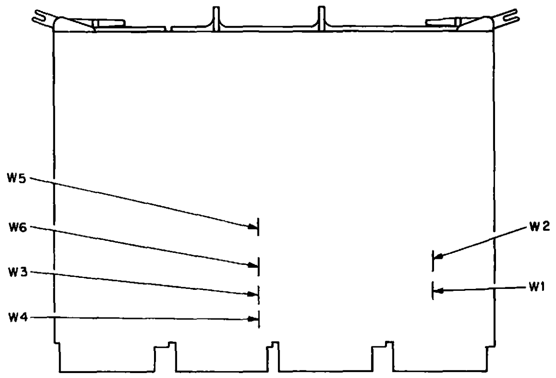
NOTE
I = Installed; R = Removed; N/A = Not applicable


607H
TYPICAL CIRCUIT SCHEMATIC REVISION IDENTIFIER (EG. CS-M7264 REV. H)
LSI11 CPU M7264 5011545D-P3 SIDE 2
TYPICAL ETCH REVISION IDENTIFIER (EG. ETCH REV. D)
A
B
C
D

Module Etch and Circuit Schematic Revision Identifiers (Module Side 2 Shown)



M7264 ETCH REV E (AND LATER)



M7264 ETCH REV. C,D (W7-W11 NOT USED)

M7264 and M7264-YA Processor Module Jumper Locations

Power-Up Mode Selection
Four power-up modes are available for user selection. These are selected (or changed) by wirewrap jumpers W5 and W6 on the processor module. Note that the jumpers affect only the power-up mode (after BDCOK H and BPOK H have been asserted); they do not affect the power-down sequence.

The state of the BHALT L signal is significant during the power-up sequence. When this signal is asserted, it invokes the processor's ODT console microcode after the power-up sequence. The console device must be properly installed for correct use of the BHALT L signal.

Power-up modes are listed below. Detailed descriptions of each mode are provided in the paragraphs that follow.

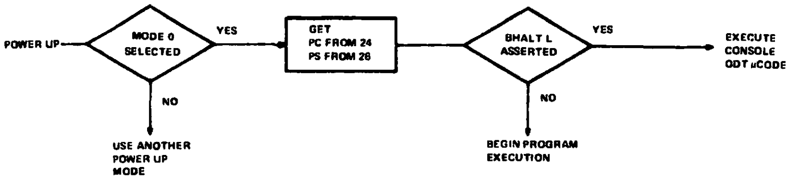
	Jumpers*		
Mode	**W6**	**W5**	**Mode Selected**
0	R	R	PC at 24 and PS at 26, or halt mode
1	R	I	ODT microde
2	I	R	PC at 173000 for user bootstrap
3	I	I	Special processor microcode (not implemented)

*R = Jumper Removed; I = Jumper Installed.

Power-Up Mode 0

This option places the processor in a microcode sequence that fetches the contents of memory locations 24 and 26 and loads their contents into the PC (R7) and the PS, respectively. A microcode service translation at this point interrogates the state of the BHALT L signal. Depending on the state of this signal, the processor either enters ODT microcode (BHALT L asserted low) or begins program execution with the current contents of R7 as the starting address (BHALT L not asserted).

Note that the T-bit (PS bit 4) is loaded with the contents of PS bit 4 in location 26. This mode should be used only with nonvolatile memory (or volatile memory with battery backup) for locations 24 and 26, or with BHALT L asserted. This power-up sequence is shown in the accompanying figure.


POWER UP
MODE 0 SELECTED
YES
NO
USE ANOTHER POWER UP MODE
GET PC FROM 24 PS FROM 26
BHALT L ASSERTED
YES
NO
EXECUTE CONSOLE ODT μCODE
BEGIN PROGRAM EXECUTION

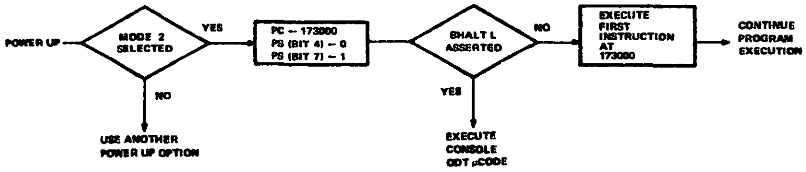
Mode 0 Power-Up Sequence

Power-Up Mode 1
This mode immediately places the processor in the console microcode regardless of the state of the BHALT L signal. This mode assumes a console interface device at bus address 177560.

Power-Up Mode 2
This mode places the processor in a microcode sequence that loads a starting address of 173000 into R7 and begins program execution at this location if the BHALT L signal is not asserted.

Note that before 173000 is loaded into R7, PS bit 4 (T-bit) is cleared and bit 7 (interrupt disable) is set. The user's program must set these bits, as desired, and set up a valid stack pointer (R6). This option should be used with nonvolatile memory (ROM, PROM, or core) at address 173000. A time-out trap through location 4 will occur if no device exists at location 173000. This mode is particularly useful when a bootstrap option is present in the system.

If BHALT L is asserted, the processor will not execute the instruction at location 173000 and will immediately execute the console microcode. This power-up mode sequence is shown in the accompanying figure.


POWER UP
MODE 2 SELECTED
YES
NO
USE ANOTHER POWER UP OPTION
PC ← 173000
PS (BIT 4) ← 0
PS (BIT 7) ← 1
BHALT L ASSERTED
NO
YES
EXECUTE CONSOLE ODT μCODE
EXECUTE FIRST INSTRUCTION AT 173000
CONTINUE PROGRAM EXECUTION

Mode 2 Power-Up Sequence

Power-Up Mode 3
This microcode sequence allows access to future microcode expansion in the fourth microm page (microlocations 3000 to 3777). After BDCOK H and BPOK H are asserted and the internal flags are cleared,

a microjump is made to microlocation 3002. If this option is selected and no microm responds to the fourth page microaddress, a microtrap will occur through microlocation 0 which will, in turn, cause a reserved user instruction trap through location 10.

Note that the state of BHALT L is not checked before control is transferred to the fourth microm page.

LTC Interrupt

Line time clock (LTC) or external event (EVNT) interrupts are enabled when jumper W3 is removed and the processor is running. The jumper can be inserted to disable this feature. The LTC interrupt is initiated by an external device when it asserts the BEVNT L signal. This is the highest priority external interrupt request; processor interrupts have higher priorities. If external interrupts are enabled (PS bit 7 = 0), the processor PC (R7) and PS word are pushed onto the processor stack. The LTC (or external event device) service routine is entered by vector address 100; the usual interrupt vector address input operation by the processor is not required since vector 100 is generated by the processor.

The first instruction of the service routine typically will be fetched within 16 μs from the time BEVNT L is asserted; however, if optional EIS/FIS instructions are being executed, this time could extend to 44.1 μs maximum. This time could also be extended by processor trap execution (memory refresh, T-bit, power fail, etc.), or by asserting the BHALT L signal.

Memory Refresh (M7264 and M7264-YA)

The LSI-11 processor has the capability of controlling the refreshing of dynamic MOS memories in a system when jumper W4 is removed. Memory refresh is *always required* when the LSI-11 system includes M7264 resident memory or MSV11-B 4K 16-bit read/write memory. The refresh operation can be controlled by a device other than the LSI-11 processor, if available, such as the REV11-A, REV11-C, and REV11-H options. If such a device is used, or if no dynamic MOS memory devices requiring "external" refresh are present in the system, install W4. The refresh sequence is described below.

The processor memory refresh sequence is controlled by resident microcode in the processor and is initiated by an internal interrupt that occurs once every 1.6 ms. It is the highest priority processor interrupt, and *cannot be disabled by software* using PS bit 7. Once the sequence is initiated, the processor will execute 64 BSYNC L/BDIN L bus transactions while asserting BREF L. The BREF L signal overrides memory bank address bits <13:15> and allows all memory units to be simultaneously enabled. After each bus transaction, BDAL<1:6> L is incremented by 1 until all 64 rows have been refreshed by the BSNYC L/BDIN L transactions. This process takes approximately 130 μs during which external interrupts (BIRQ L and BEVNT L) are ignored. However, DMA requests can be granted between each of the 64 refresh transactions.

WRITABLE CONTROL STORE

The LSI-11 Writable Control Store (WCS) option consists of an LSI-11 CPU and a WCS module that provides up to 1K (1024) words of user programmable microcode. A standard LSI-11 processor module contains two Microcode Read Only Memory (MICROM) chips that define the PDP-11 instruction set. An extra MICROM socket on the LSI-11 processor may optionally contain the KEV11 Extended Arithmetic MICROM or a connecting cable from the WCS module.

The additional microcode space available permits expansion of the PDP-11 instruction set. WCS is most useful where an application has a well-defined function causing a performance bottleneck. After such a function has been microcoded, data access time and processing time are significantly reduced. For the sophisticated user, WCS can be a cost-effective alternative to a more powerful and expensive processor.

The WCS module contains Read/Write memory chips which can be loaded through an LSI-11 system bus interface and then locked into a read-only mode. The WCS memory chips are volatile and must be reloaded each time the system is powered up.

The LSI-11 has a vertical microcode structure which resembles assembly language programming. Optional software tools are available for use with the RT-11 operating system to facilitate microcode development.

Software Development Tools

Software development tools for the LSI-11 WCS are available for use with RT-11 V3 systems having a minimum 16K words of memory. The software tools are distributed on a floppy disk and include a micro-assembler, à micro-loader, micro-debugging aid, and a WCS microcode Save/Recall routine. User documentation for the software tools is contained in a single volume, designed for ease of use by the reader with LSI-11 experience. The software tools package is available under a Category C software license (implying no direct software support from DIGITAL) and requires that the user have a Catgory A (full support) RT-11 software license. Sources and binaries of the tools are distributed. The package also includes the sources of the KEV11 microcode for study or use.

Microcode Applications

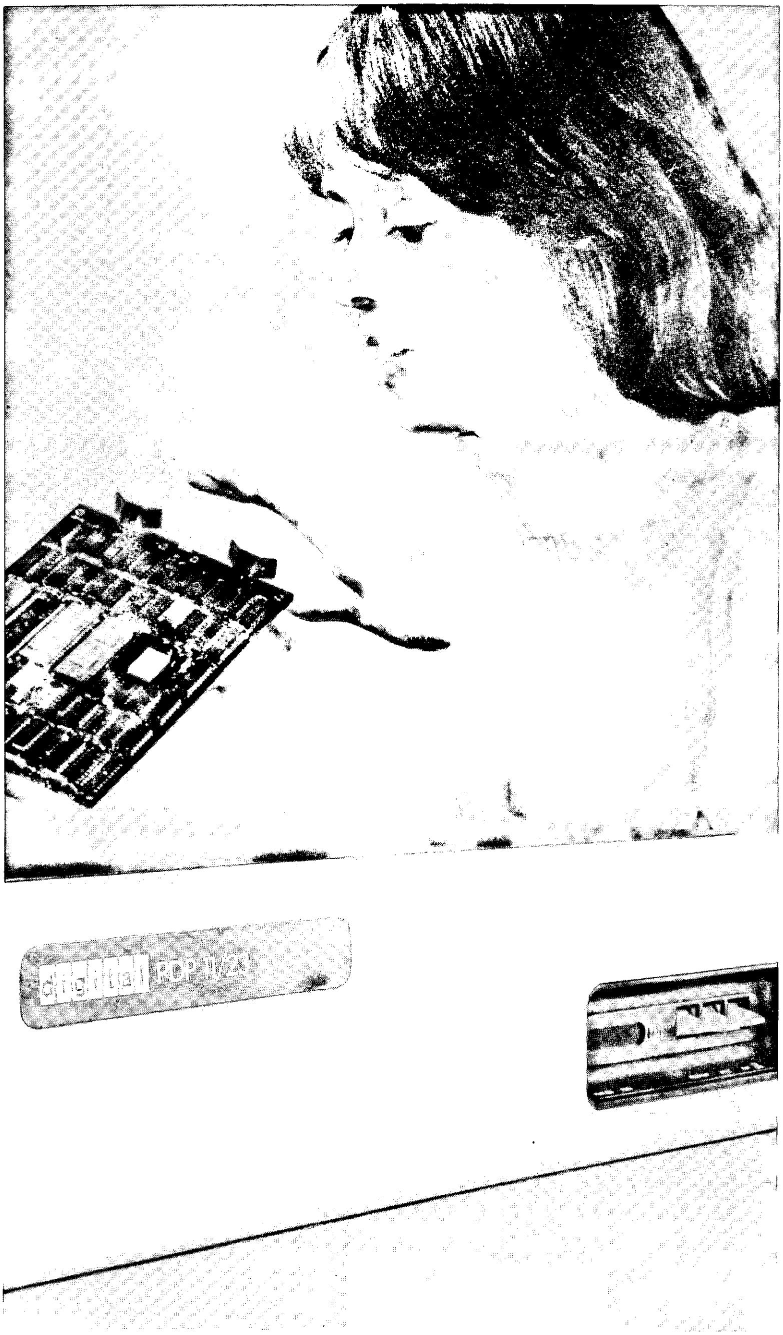
- Arithmetic Calculations—Many arithmetic calculations are characterized by concisely defined, often repetitive, algorithms. Microcoding of such algorithms will yield substantial time savings by eliminating machine instruction fetches and operand address calculations. An entire algorithm can be microcoded so as to execute in response to a single, user-defined machine instruction.
- Time Critical I/O Operations—The rate at which real-time I/O operations can be performed depends upon the speed of machine instruction execution and the speed of the LSI-11 bus. The microprogramming facility allows specialized I/O routines to be written in microcode to make optimal use of system resources. Processing and I/O operations can be interleaved in a manner impossible using the higher-level PDP-11 machine instructions.
- Data Manipulation—Routines which perform data manipulation and relocation within memory can be highly optimized in microcode. For example, in the case of a block move operation, time saved is proportional to block length because separate machine instruction fetches are eliminated for each word moved.

WCS FEATURES

- 1K X 24-bit microcode RAM
- Two additional Special Function bits in each microcode word are available as TTL signals on the backplane. These two bits plus four Special Function bits on the LSI-11 CPU may be used under microcode control to generate high-speed control signals for external hardware.
- A hardware trace RAM on the WCS module displays the microcode sequence executed up to a user defined stop-trace point. This aids in microcode development and is supported by the optional WCS software tools.

SPECIFICATIONS

Operating Environment:	5°–60° C (40°–140°F) 10% to 95% relative humidity (noncondensing)
Power:	+5.0 V ±5%
Typical Current:	3.0 A


digital PDP 11/23

CHAPTER 6

ADDRESSING MODES

In the LSI-11 and PDP-11 family, all memory reference addressing is accomplished using the eight general purpose registers. In specifying an address of the data (operand address), one of the eight registers is selected and one of several addressing modes. Each memory reference instruction specifies the:

- function to be performed (operation code)
- general purpose register to be used when locating the source destination and/or destination operand
- addressing mode, which specifies how the selected registers are to be used

The instruction format and addressing techniques available to the programmer are of particular importance. It is the combination of addressing modes with the instruction set that provides the 11 family a unique number of capabilities. The LSI-11 and the PDP-11 are designed to handle structured data efficiently and with flexibility. The general purpose registers implement these functions in the following ways, by acting:

- as accumulators: holding the data to be manipulated
- as pointers: the contents of the register are the address of the operand, rather than the operand itself, allowing automatic stepping through memory locations.
- as index registers: the contents of the register are added to the second word of the instruction to produce the address of the operand. This capabilty allows easy access to variable entries in a list.

Utilization of the registers for both data manipulation and address calculation results in a variable length instruction format. If registers alone are used to specify the data source, only one memory word is required to hold the instruction. In certain modes, two or three words may be utilized to hold the basic instruction components. Special addressing mode combinations enable temporary data storage for convenient dynamic handling of frequently accessed data. This is known as **stack addressing**. Programming techniques utilizing the stack are discussed in Chapter 11. Register 6 is always used as the hardware stack pointer, or SP. Register 7 is used by the processor as its program counter (PC). Thus, the register arrangement to be considered in conjunction with instructions and with addressing modes is: registers 0-5 are general purpose registers, register 6 is the hardware stack pointer, and register 7 is the program counter. The full instruction set and instruction formats are explained in Chapter 7.

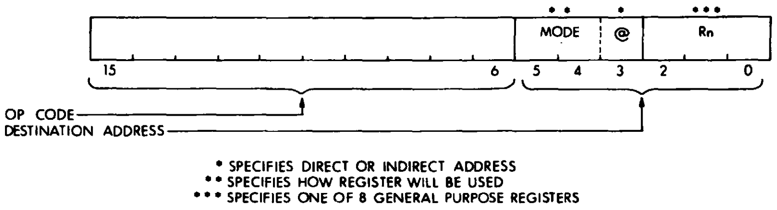
For the purpose of clearly illustrating the use of the various addressing modes, the following instructions are used in this chapter:

Mnemonic	Description	Octal Code
CLR	Clear (Zero the specified destination.)	0050DD
CLRB	Clear Byte (Zero the byte in the specified destination.)	1050DD
INC	Increment (Add 1 to contents of destination.)	0052DD
INCB	Increment Byte (Add 1 to the contents of destination byte.)	1052DD
COM	Complement (Replace the contents of the destination by their logical 1's complements; each 0 bit is set and each 1 bit is cleared.)	0051DD
COMB	Complement Byte (Replace the contents of the destination byte by their logical 1's complements; each 0 bit is set and each 1 bit is cleared.)	1051DD
ADD	Add (Add source operand to destination operand and store the result at destination address.)	06SSDD

DD = destination field (6 bits)
SS = source field (6 bits)
() = contents of

Single and double operand instructions utilize the following format.

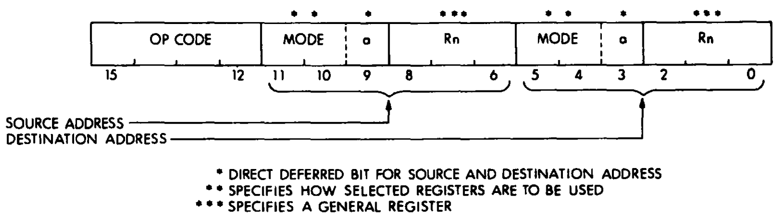
The instruction format for the first word of all single operand instructions (such as clear, increment, test) is:



MODE
@
Rn
15
6
5
4
3
2
0
OP CODE
DESTINATION ADDRESS
* SPECIFIES DIRECT OR INDIRECT ADDRESS
** SPECIFIES HOW REGISTER WILL BE USED
*** SPECIFIES ONE OF 8 GENERAL PURPOSE REGISTERS

Single Operand Instruction Format

The instruction format for the first word of the double operand instruction is as follows:



OP CODE
MODE
a
Rn
MODE
a
Rn
15
12
11
10
9
8
6
5
4
3
2
0
SOURCE ADDRESS
DESTINATION ADDRESS
* DIRECT DEFERRED BIT FOR SOURCE AND DESTINATION ADDRESS
** SPECIFIES HOW SELECTED REGISTERS ARE TO BE USED
*** SPECIFIES A GENERAL REGISTER

Double Operand Instruction Format

Bits 3-5 specify the binary code of the addressing mode chosen.

The four direct addressing modes are:

- register
- autoincrement
- autodecrement
- index

When bit 3 of the instruction is set, indirect addressing is specified and the four basic modes become deferred modes. In a register deferred mode, the content of the selected register is taken as the address of the operand. In the other deferred modes, the content of the register specifies the address of the operand, rather than the operand itself. Prefacing the register operand(s) with an "@" sign or placing the register in parentheses indicates to the MACRO-11 assembler that deferred addressing mode is being used.

The indirect addressing modes are:

- register deferred
- autoincrement deferred
- autodecrement deferred
- index deferred

Program counter (register 7) addressing modes are:

- immediate
- absolute
- relative
- relative deferred

The addressing modes are explained and shown in examples in the following pages. They are summarized, in text and in graphic representation, at the end of the chapter.

REGISTER MODE **MODE 0** **Rn**

Register mode provides faster instruction execution. There is no need to reference memory to retrieve an operand. Any of the general registers can be used as simple accumulators. The operand is contained in the selected register. Assembler syntax requires that a general register be defined as follows:

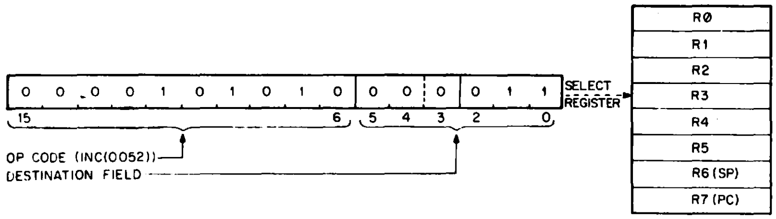
R0 = %0
R1 = %1
R2 = %2

% sign indicates register definition.

Register Mode Example

Symbolic	Instruction Octal Code	Description
INC R3	005203	Add 1 to the contents of R3.

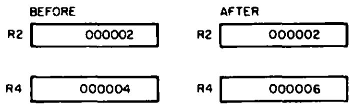
Represented as:



Register Mode Example

Symbolic	Instruction Octal Code	Description
ADD R2,R4	060204	Add the contents of R2 to the contents of R4, replacing the original contents of R4 with the sum.

Represented as:



REGISTER DEFERRED MODE **MODE 1** **(Rn)**

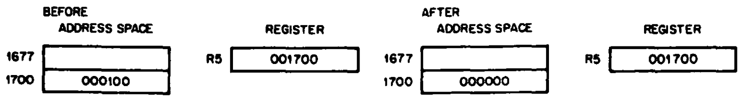
In register deferred mode, the address of the operand is stored in a general purpose register. The address contained in the general purpose register directs the CPU to the operand. The operand is located outside the CPU, either in memory, or in an I/O register.

This mode is used for: sequential lists, indirect pointers in data structures, top of stack manipulations, and jump tables.

Register Deferred Mode Example

Symbolic	Instruction Octal Code	Description
CLR (R5)	005015	The contents of the location specified in R5 are cleared.

Represented as:



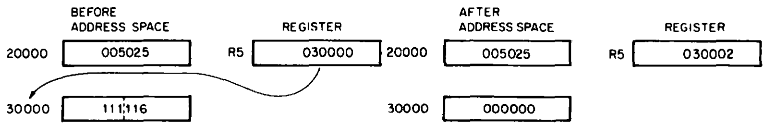
AUTOINCREMENT MODE **MODE 2** **(Rn)+**

In autoincrement mode, the register contains the address of the operand; the address is automatically incremented after the operand is retrieved. The address then references the next sequential operand. This mode allows automatic stepping through a list or series of operands stored in consecutive locations. When an instruction calls for mode 2, the address stored in the register is autoincremented each time the instruction is executed. It is autoincremented by 1 if you are using byte instructions, by 2 if you are using word instructions.

Autoincrement Mode Example

Symbolic	**Instruction Octal Code**	**Description**
CLR (R5)+	005025	Contents of R5 are used as the address of the operand. Clear selected operand and then increment the contents of R5 by 2.

Represented as:



AUTOINCREMENT DEFERRED MODE   MODE 3   @(Rn)+

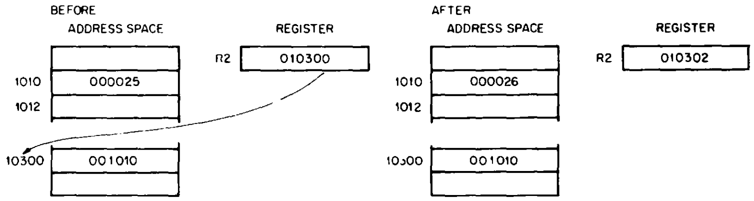
In autoincrement deferred mode, the register contains a pointer to an address. The "+" indicates that the pointer in R2 is incremented by 2 after the address is located. Mode 2, autoincrement, is used only to access operands that are stored in consecutive locations. Mode 3, autoincrement deferred, is used to access lists of operands stored anywhere in the system; i.e., the operands do not have to reside in adjoining locations. Mode 2 is used to step through a table of volumes, mode 3 is used to step through a table of addresses.

Autoincrement Deferred Example

Symbolic	**Instruction Octal Code**	**Description**
INC @(R2)+	005232	Contents of R2 are used as the address of the address of the operand. The operand is increased by 1,

contents of R2 are incremented by 2.

Represented as:



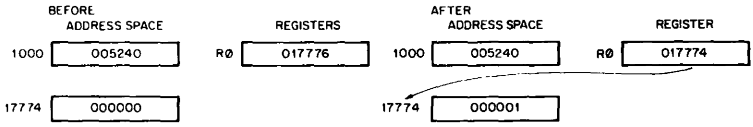
AUTODECREMENT MODE — MODE 4 — −(Rn)

In autodecrement mode, the register contains an address that is automatically decremented; the decremented address is used to locate an operand. This mode is similar to autoincrement mode, but allows stepping through a list of words or bytes in reverse order. The address is autodecremented by 1 for bytes, by 2 for words.

Autodecrement Mode Example

Symbolic	Instruction Octal Code	Description
INCB −(R0)	105240	The contents of R0 are decremented by 1, then used as the address of the operand. The operand byte is increased by 1.

Represented as:



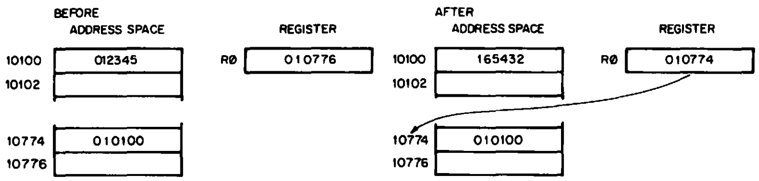
AUTODECREMENT DEFERRED MODE — MODE 5 — @−(Rn)

In autodecrement deferred mode, the register contains a pointer. The pointer is first decremented by 2, then the new pointer is used to retrieve an address stored outside the CPU. This mode is similar to autoincrement deferred, but allows stepping through a table of addresses in reverse order. Each address then redirects the CPU to an operand. Note that the operands do not have to reside in consecutive locations.

Autodecrement Deferred Mode Example

Symbolic	Instruction Octal Code	Description
COM @-(R0)	005150	The contents of R0 are decremented by 2 and then used as the address of the address of the operand. The operand is 1's complemented.

Represented as:



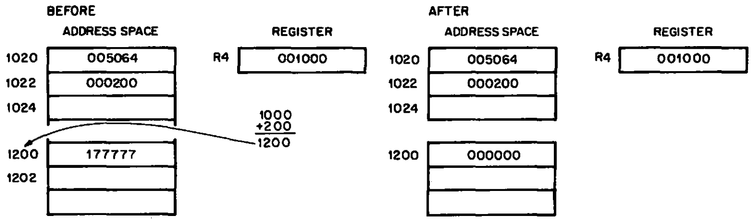
INDEX MODE **MODE 6** **±X(Rn)**

In index mode, a base address is added to an index word to produce the effective address of an operand; the base address specifies the starting location of table or list. The index word then represents the address of an entry in the table or list relative to the starting (base) address. The base address may be stored in a register. In this case, the index word follows the current instruction. Or the locations of the base address and index word may be reversed (index word in the register, base address following the current instruction).

Index Mode Example

Symbolic	Instruction Octal Code	Description
CLR 200(R4)	005064 000200	The address of the operand is determined by adding 200 to the contents of R4. The location is then cleared.

Represented as:



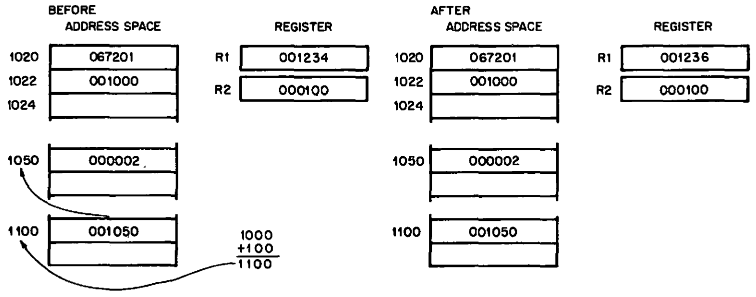
INDEX DEFERRED MODE — MODE 7 — @X(Rn)

In index deferred mode, a base address is added to an index word. The result is a pointer to an address, rather than the actual address. This mode is similar to mode 6, except that it produces a pointer to an address. The content of that address then redirects the CPU to the desired operand. Mode 7 provides for the random access of operands using a table of operand addresses.

Index Deferred Mode Example

Symbolic	Instruction Octal Code	Description
Add @1000(R2),R1	067201 001000	1000 and the contents of R2 are summed to produce the address of the address of the source operand, the contents of which are added to the contents of R1. The result is stored in R1.

Represented as:



USE OF THE PC AS A GENERAL REGISTER

Register 7 is both a general purpose register and the program counter on the LSI-11 and the PDP-11. When the CPU uses the PC to access a word from memory, the PC is automatically incremented by two to contain the address of the next word of the instruction being executed or the address of the next instruction to be executed. When the program uses the PC to access byte data, the PC is still incremented by two.

The PC can be used with all the 11 addressing modes. There are four modes in which the PC can provide advantages for handling position-independent code and for handling unstructured data. These modes refer to the PC and are termed immediate, absolute (or immediate deferred), relative, and relative deferred.

PC IMMEDIATE MODE — MODE 2 — #n

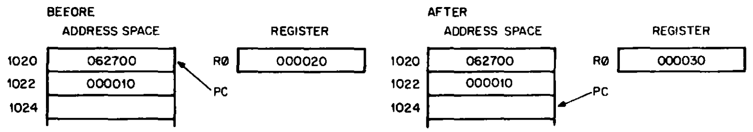
Immediate mode is equivalent to using the autoincrement mode with the PC. It provides time improvements for accessing constant operands by including the constant in the memory location immediately following the instruction word.

PC Immediate Mode Example

Symbolic	Instruction Octal Code	Description
ADD #10,R0	062700 000010	The value 10 is located in the second word of the instruction and is added to the contents of R0. Just before this instruction is fetched and executed, the PC points to the first word of the instruction. The processor fetches the first word and increments the PC by two. The source operand mode is 27 (autoincrement the PC). Thus, the PC is used as a pointer to fetch the operand (the second word of the in-

struction) before being incremented by two to point to the next instruction.

Represented as:



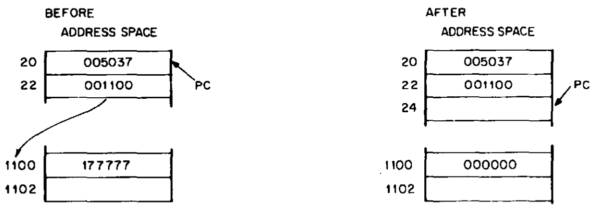
PC ABSOLUTE MODE **MODE 3** **@#A**

This mode is the equivalent of immediate deferred or autoincrement deferred mode using the PC. The contents of the location following the instruction are taken as the address of the operand. Immediate data is interpreted as an absolute address (i.e., an address that remains constant no matter where in memory the assembled instruction is executed).

PC Absolute Mode Example

Symbolic	**Instruction Octal Code**	**Description**
CLR @#1100	005037 001100	Clears the contents of location 1100.

Represented as:



PC RELATIVE MODE **MODE 6** **A**

This mode is index mode 6 using the PC. The operand's address is calculated by adding the word that follows the instruction (called an "offset") to the updated contents of the PC.

PC+2 directs the CPU to the offset that follows the instruction. PC+4 is summed with this offset to produce the effective address of the operand. PC+4 also represents the address of the next instruction in the program.

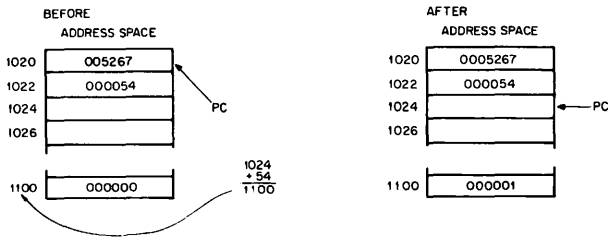
With the relative addressing mode, the address of the operand is always determined with respect to the updated PC. Therefore, when the instruction is relocated, the operand remains the same relative distance away.

The distance between the updated PC and the operand is called an **offset**. After a program is assembled, this offset appears in the first word location that follows the instruction. This mode is useful for writing position-independent code.

PC Relative Mode Example

Symbolic	**Instruction Octal Code**	**Description**
INC A	005267 000054	To increment location A, contents of memory location in the second word of the instruction are added to PC to produce address A. Contents of A are increased by 1.

Represented as:



PC RELATIVE DEFERRED MODE MODE 7 @A

This mode is index deferred (mode 7), using the PC. A pointer to an operand's address is calculated by adding an offset (that follows the instruction) to the updated PC.

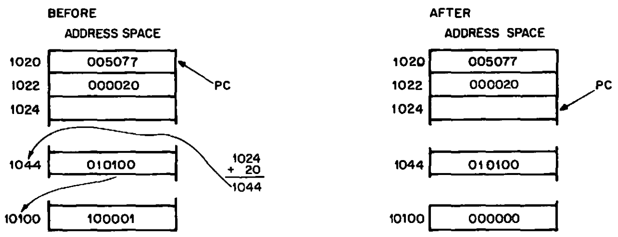
This mode is similar to the relative mode, except that it involves one additional level of addressing to obtain the operand. The sum of the offset and updated PC (PC+4) serves as a pointer to an address. When the address is retrieved, it can be used to locate the operand.

PC Relative Deferred Mode Example

Symbolic	**Instruction Octal Code**	**Description**
CLR @A	005077	Adds the second

	000020	word of the instruction to PC to produce the address of the address of the operand. Clears operand.

Represented as:



Direct Addressing Modes

Binary Code	Mode	Name	Symbolic	Function
000	0	Register	Rn	Register contains operand.
010	2	Autoincrement	(Rn)+	Register is used as a pointer to sequential data, then incremented.
100	4	Autodecrement	−(Rn)	Register is decremented and then used as a pointer to sequential data.
110	6	Index	X(Rn)	Value X is added to (Rn) to produce address of operand. Neither X nor (Rn) is modified.

Indirect Addressing Modes

Binary Code	**Mode**	**Name**	**Symbolic**	**Function**
001	1	Register Deferred	@Rn or (Rn)	Register contains the address of the operand.
011	3	Autoincrement Deferred	@(Rn)+	Register is first used as a pointer to a word containing the address of the operand, then incremented (always by 2, **even** for byte instructions).
101	5	Autodecrement Deferred	@−(Rn)	Register is decremented (always by 2, **even** for byte instructions) and then used as a pointer to a word containing the address of the operand.
111	7	Index Deferred	@X(rn)	Value X (located in a word contained in the instruction) and (Rn) are added and the sum is used as a pointer to a word containing the address of the operand. Neither X nor (Rn) is modified.

When used with the PC, these modes are termed inmmediate, absolute (or immediate deferred), relative, and relative deferred.

PC Register Addressing Modes

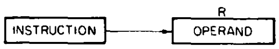
Binary Code	**Mode**	**Name**	**Symbolic**	**Function**
010	2	Immediate	#n	Operand is contained in the instruction.
011	3	Absolute	@#A	Absolute address is contained in the instruction.
110	6	Relative	A	Address of A, relative to the instruction, is contained in the instruction.
111	7	Relative Deferred	@A	Address of location containing address of A, relative to the instruction, is contained in the instruction.

GRAPHIC SUMMARY OF PDP-11 ADDRESSING MODES

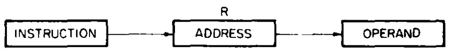
General Register Addressing Modes

R is a general register, 0 to 7.
(R) is the contents of that register.

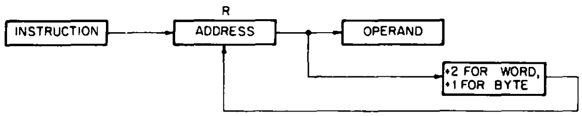
Mode 0	**Register**	OPR R	R contains operand.



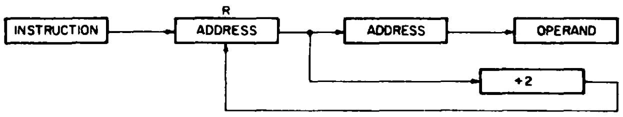
Mode 1 **Register deferred** OPR (R) R contains address.



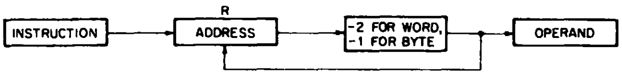
Mode 2 **Autoincrement** OPR (R)+ R contains address, then increment (R).



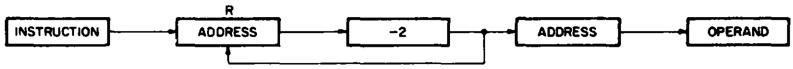
Mode 3 **Autoincrement deferred** OPR @(R)+ R contains address of address, then increment (R) by 2.



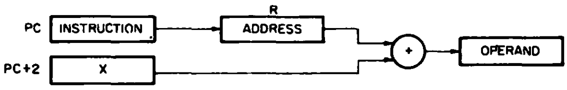
Mode 4 **Autodecrement** OPR −(R) Decrement (R), then R contains address.



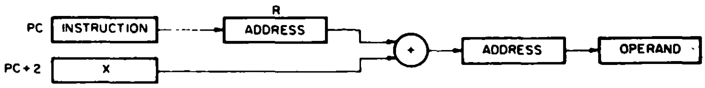
Mode 5	**Autodecrement deferred**	OPR @–(R)	Decrement (R) by 2, then R contains address of address.



Mode 6	**Index**	OPR X(R)	(R)+X is address, second word of instruction.

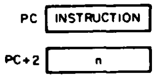


Mode 7	**Index deferred**	OPR @X(R)	(R)+X is address (second word) of address.

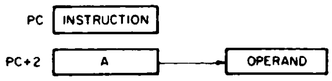


Program Counter Addressing Modes
Register = 7

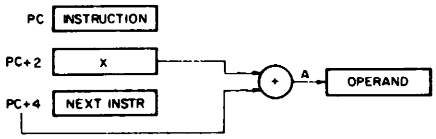
Mode 2	**Immediate**	OPR #n	Literal operand n is contained in the instruction.



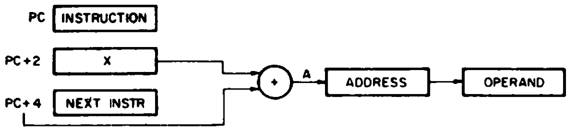
Mode 3	**Absolute**	OPR @#A	Address A is contained in the instruction.

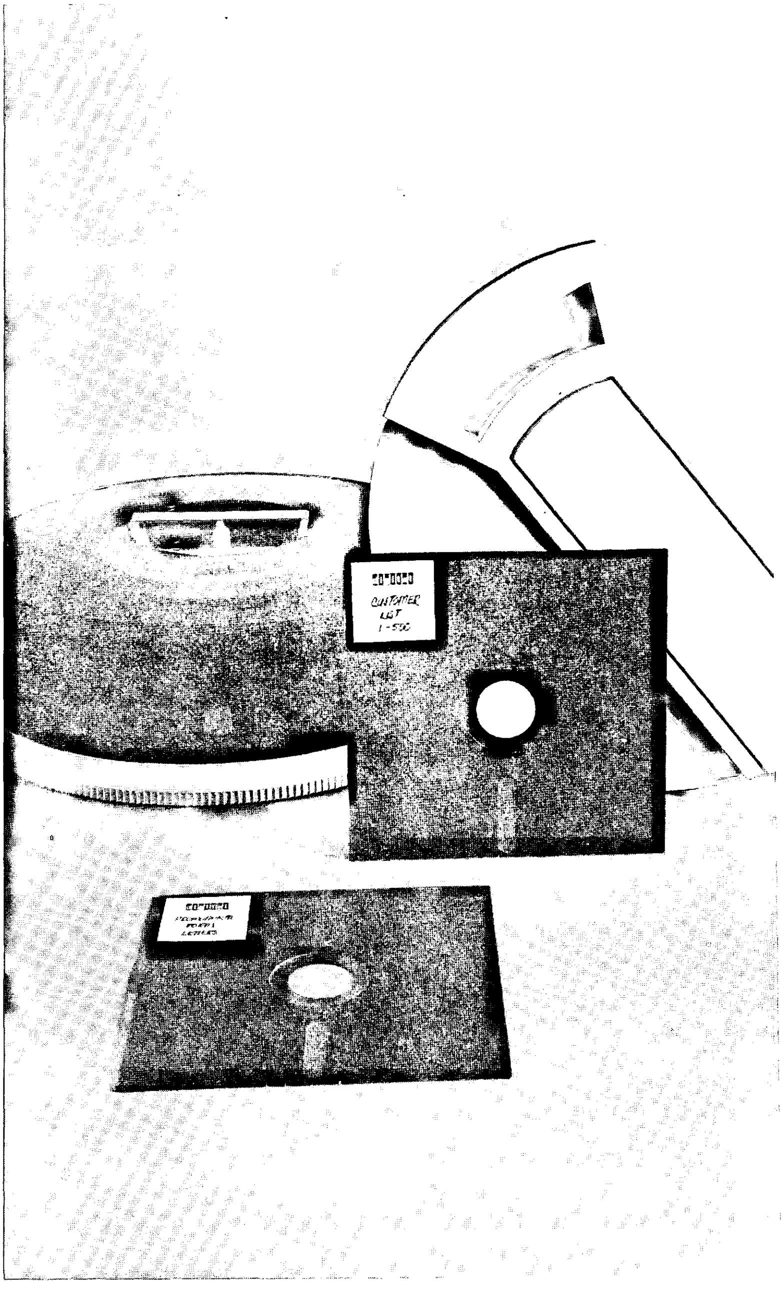


Mode 6	**Relative**	OPR A	PC+4 + X is address. PC+4 is updated PC.



Mode 7	**Relative deferred**	OPR @A	PC+4 + X is address of address. PC+4 is updated PC.




CUSTOMER LIST 1-500

CHAPTER 7
INSTRUCTION SET

The 11 instruction set and addressing modes produce over 400 unique instructions. The instruction set offers a wide choice of operations, so that a single instruction will frequently accomplish a task that would require several in a traditional computer. PDP-11 instructions allow byte and word addressing in both single and double operand formats. This saves memory space and simplifies the implementation of control and communications applications. The PDP-11's use of double operand instructions allows you to perform several operations with a single instruction. For example, ADD A,B adds the contents of location A to location B, storing the result in location B. Traditional computers would implement this instruction in the following way:

```
CLR A,C
LDA A
ADD B
STR B
```

The 11 instruction set also contains a full set of conditional branches, eliminating excessive use of jump instructions. All PDP-11 instructions fall into one of three categories:

- *Single Operand* — one part of the word specifies the operation, referred to as "op code," the second part provides information for locating the operand.
- *Double Operand* — the first part of the word specifies the operation to be performed, the remaining two parts provide information for locating two operands.
- *Program Control* — the first part of the word specifies the operation to be performed, the second part indicates where the action is to take place in the program.

SINGLE OPERAND INSTRUCTIONS

	Mnemonic	**Instruction**
General		
	CLR(B)	clear destination
	COM(B)	1's complement dst
	INC(B)	increment dst
	DEC(B)	decrement dst
	NEG(B)	2's complement negate dst
	TST(B)	test dst

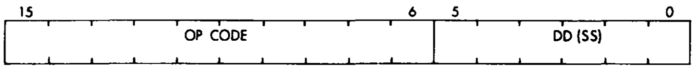
Shift & Rotate

	ASR(B)	arithmetic shift right
	ASL(B)	arithmetic shift left
	ROR(B)	rotate right
	ROL(B)	rotate left
	SWAB	swap bytes

Multiple Precision

	ADC(B)	add carry
	SBC(B)	subtract carry
	SXT	sign extend
	MFPS	move byte from processor status
	MTPS	move byte to processor status

Instruction Format



Single Operand Instruction Format

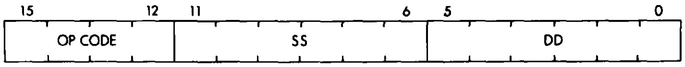
The instruction format for single operand instructions is:

- Bit 15 indicates word or byte operation.
- Bits 14-6 indicate the operation code, which specifies the operation to be performed.
- Bits 5-0 indicate the 3-bit addressing mode field and the 3-bit general register field. These two fields are referred to as the destination field.

DOUBLE OPERAND INSTRUCTIONS

	Mnemonic	**Instruction**
General		
	MOV(B)	move source to destination
	ADD	add src to dst
	SUB	subtract src from dst
	ASH	shift arithmetically
	ASHC	arithmetic shift combined
Logical		
	BIT(B)	bit test
	BIC(B)	bit clear
	BIS(B)	bit set
	XOR	exclusive OR

Instruction Format



Double Operand Instruction Format

The format of most double operand instructions is similar to that of single operand instructions except that they have *two* fields for locating operands. One field is called the source field, the other is called the destination field. Each field is further divided into addressing mode and selected register. Each field is completely independent. The mode and register used by one field may be completely different than the mode and register used by another field.

- Bit 15 indicates word or byte operation *except* when used with op code 6. Then it indicates an ADD or SUBtract instruction.
- Bits 14-12 indicate the op code, which specifies the operation to be done.
- Bits 11-6 indicate the 3-bit addressing mode field and the 3-bit general register field. These two fields are referred to as the **source** field.
- Bits 5-0 indicate the 3-bit addressing mode field and the 3-bit general register field. These two fields are referred to as the **destination** field.

Byte Instructions
Byte instructions are specified by setting bit 15. Thus, in the case of the MOV instruction, bit 15 is 0; when bit 15 is set, the mnemonic is MOVB. There are no byte operations for ADD and SUB, i.e., no ADDB or SUBB.

PROGRAM CONTROL INSTRUCTIONS

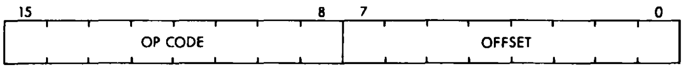
Branch Instructions

	Mnemonic	Instruction
Branch		
	BR	branch (unconditional)
	BNE	branch if not equal (to zero)
	BEQ	branch if equal (to zero)
	BPL	branch if plus
	BMI	branch if minus
	BVC	branch if overflow is clear
	BVS	branch if overflow is set
	BCC	branch if carry is clear
	BCS	branch if carry is set

Branch Instructions

	Mnemonic	Instruction
Signed Conditional Branch		
	BGE	branch if greater than or equal (to zero)
	BLT	branch if less than (zero)
	BGT	branch if greater than (zero)
	BLE	branch if less than or equal (to zero)
	SOB	subtract one and branch (if not = 0)
Unsigned Conditional Branch		
	BHI	branch if higher
	BLOS	branch if lower or same
	BHIS	branch if higher or same
	BLO	branch if lower

Instruction Format



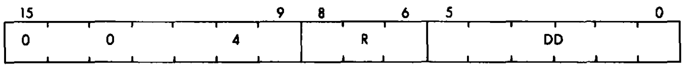
Branch Instruction Format

- The high byte (bits 8-15) of the instruction is an op code specifying the conditions to be listed.
- The low byte (bits 0-7) of the instruction is the offset value in words that determines the new program location if the branch is taken.

JUMP AND SUBROUTINE INSTRUCTIONS

	Mnemonic	Instruction
Jump & Subroutine		
	JMP	jump
	JSR	jump to subroutine
	RTS	return from subroutine

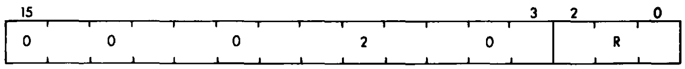
Instruction Format
JSR Format



JSR Instruction Format

- Bits 9-15 are always octal 004 indicating the op code for JSR.
- Bits 6-8 specify the link register. Any general purpose register may be used in the link, except R6.
- Bits 0-5 designate the destination field that consists of addressing mode and general register fields. This specifies the starting address of the subroutine.
- Register R7 (The Program Counter) is frequently used for both the link and the destination. For example, you may use JSR R7, SUBR, which is coded 004767. R7 is the *only* register that can be used for both the link and destination, the other GPRs cannot. Thus, if the link is R5, any register except R5 can be used for one destination field.

RTS Format



RTS Instruction Format

The RTS (return from subroutine) instruction uses the link to return control to the main program once the subroutine is finished.

- Bits 3-15 always contain octal 00020, which is the op code for RTS.
- Bits 0-2 specify any one of the general purpose registers.
- The register specified by bits 0-2 must be the same register used as the link between the JSR causing the jump and the RTS returning control.

INTERRUPTS AND TRAPS

Mnemonic	**Instruction**
EMT	emulator trap
TRAP	trap
BPT	breakpoint trap
IOT	input/output trap
RTI	return from interrupt
RTT	return from interrupt

There are three ways of leaving a main program:

- *software exit* — the program specifies a jump to some subroutine
- *trap exit* — internal hardware on a special instruction forces a jump to an error handling routine
- *interrupt exit* — external hardware forces a jump to an interrupt service routine

In all of the above cases, there is a jump to another program. Once that program has been executed, control is returned to the proper point in the main program.

MISCELLANEOUS INSTRUCTIONS

Mnemonic	Instruction
HALT	halt
WAIT	wait for interrupt
RESET	reset UNIBUS
MTPD	move to previous data space
MTPI	move to previous instruction space
MFPD	move from previous data space
MFPI	move from previous instruction space
MTPS	move byte to processor status word
MFPS	move byte from processor status word

CONDITION CODE OPERATION

Mnemonic	Instruction
CLC,CLV,CLZ,CLN,CCC	clear
SEC,SEV,SEZ,SEN,SCC	set

There are four condition code bits:

- N, indicating a negative condition when set to 1
- Z, indicating a zero condition when set to 1
- V, indicating an overflow condition when set to 1
- C, indicating a carry condition when set to 1

These four bits are part of the processor status word (PS). The result of any single operand or double operand instruction affects one or more of the four condition code bits. A new set of condition codes is usually created after execution of each instruction. Some condition codes are not affected by the execution of certain instructions. The CPU may be asked to check the condition codes after execution of an instruction. The condition codes are used by the various instructions to check software conditions.

Z bit — Whenever the CPU sees that the result of an instruction is zero, it sets the Z bit. If the result is not zero, it clears the Z bit. There are a number of ways of obtaining a zero result:

- adding two numbers equal in magnitude but different in sign
- comparing two numbers of equal value
- using the CLR instruction

N bit — The CPU looks only at the sign bit of the result. If the sign bit is set, indicating a negative value, the CPU sets the N bit. If the sign bit is clear, indicating a positive value, then the CPU clears the N bit.

C bit — The CPU sets the C bit automatically when the result of an instruction has caused a carry out of the most significant bit of the result. When the instruction results in a carry out of the most significant bit of the result, the carry itself is usually moved into the C bit. Otherwise, the C bit is cleared. During rotate instructions (ROL and ROR), the C bit forms a buffer between the most significant bit and the least significant bit of the word. A carry of 1 sets the C bit while a carry of 0 clears the C bit. However, there are exceptions. For example:

- SUB and CMP set the C bit when there is no carry.
- INC and DEC do not affect the C bit.
- COM always sets the C bit, TST always clears the C bit.

V bit — The V bit is set to indicate that an overflow condition exists. An overflow means that the result of an instruction is too large to be placed in the destination. There are two methods the hardware uses to check for an overflow condition.

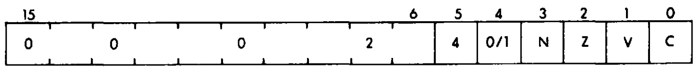
One way is for the CPU to test for a change of sign.

- When using single operand instructions, such as INC, DEC, or NEG, a change of sign indicates an overflow condition.
- When using double operand instructions, such as ADD, SUB, or CMP, in which both the source and destination have like signs, a change of sign in the result indicates an overflow condition.

Another method used by the CPU is to test the N bit and C bit when dealing with shift and rotate instructions.

- If only the N bit is set, an overflow exists.
- If only the C bit is set, an overflow exists.
- If *both* the N and C bits are set, there is no overflow condition.

More than one condition code can be set by a particular instruction. For example, both a carry and an overflow condition may exist after instruction execution.



Condition Code Operators' Format

Instruction Format

The format of the condition code operators is as follows:

- Bits 15-5 — the "op" code
- Bit 4 — the "operator" which indicates set or clear with the values 1 and 0 respectively. If set, any selected bit is set; if clear, any selected bit is cleared.
- Bits 3-0 — the "select" field. Each of these bits corresponds to one of the four condition code bits. When one of these bits is set, then the corresponding condition code bit is set or cleared depending on the state of the "operator" (bit 4).

EXAMPLES

The following examples and explanations illustrate the use of the various types of instructions in a program.

Single Operand Instruction Example

This routine uses a tally to control a loop, which clears out a specific block of memory. The routine has been set up to clear 30_8 byte locations beginning at memory address 600.

```
(R0) = 600
(R1) = 30
LOOP:       CLRB(R0)+
            DEC R1
            BNE R1
            LOOP
            HALT
```

Program Description

- The CLRB (R0)+ instruction clears the content of the location specified by R0 and increments R1.
- R0 is the pointer.
- Because the auto-increment addressing mode is used, the pointer automatically moves to the next memory location after execution of the CLRB instruction.
- Register R1 indicates the number of locations to be cleared and is, therefore, a counter. Counting is performed by the DEC R1 instruction. Each time a location is cleared, it is counted by decrementing R1.
- The Branch If Not Zero, BNE, instruction checks for done. If the counter is not zero, the program branches back to start to clear another location. If the counter is zero, indicating done, then the program executes the next instruction, HALT.

Double Operand Instruction Example

This routine prints out a portion of a payroll program for review by the supervisor. It is known that 76 locations are to be printed and the locations start at address 600.

```
INIT:         MOV #600, R0
              MOV #76, R1

START:        MOVB (R0)+, I/O
              DEC R1
              BNE START
              HALT
```

Program Description

- MOV is the instruction normally used to set up the initial conditions. Here, the first MOV places the starting address (600) into R0, which will be used as a *pointer*. The second MOV sets up R1 as a *counter* by loading the desired number of locations (76) to be printed.
- The MOVB instruction moves a byte of data to the printer (I/O) for printing. The data comes from the location specified by R0. The pointer R0 is then incremented to point to the next sequential location.
- The counter (R1) is then decremented to indicate one byte has been transferred.
- The program then checks the loops for done with the BNE instruction. If the counter has not reached zero, indicating more transfers must take place, then the BNE causes a branch back to START and the program continues.
- When the counter (R1) reaches zero, indicating all data has been transferred, the branch does not occur and the program executes the next instruction, HALT.

Branch Instruction Example

NOTE

Branch instructions are limited from $+177_8$ to -200_8 words.

A payroll program has set up a series of words to identify each employee by his badge number. The high byte of the word contains the employee's badge number, the low byte contains an octal number ranging from 0 to 13 which represents his salary. These numbers represent steps within three wage classes to identify which employees get paid weekly, monthly, or quarterly. It is time to make out weekly paychecks. Unfortunately, employee information has been stored in a random order. The problem is to extract the names of only those employees who receive a weekly paycheck. Employee payroll numbers are assigned as follows: 0 to 3 — Wage Class I (weekly), 4 to 7 — Wage Class II (monthly), 10 to 13 — Wage Class III (quarterly).

600 is the starting address of memory block containing the employee payroll information. 1264 is the final address of this data area. The following program searches through the data area and finds all numbers representing wage class I, and, each time an appropriate number is found, stores the employee's badge number (just the high byte) on a "last-in/first-out" stack which begins at location 400.

```
INIT:       MOV #600, R0
            MOV #400, R1

START:      CMPB(R0)+,#3


            BHI CONT

STACK:      MOVB (R0),-(R1)

CONT:       INC R0

            CMP #1264, R0

            BHIS START

            HALT
```

Program Description

- R0 becomes the address pointer, R1 the stack pointer.
- Compare the contents of the first low byte with the number 3 and go to the first high byte.
- If the number is more than 3, branch to continue.
- If no branch occurs, it indicates that the number is 3 or less. Therefore, move the high byte containing the employee's number onto the stack as indicated by stack pointer R1.
- R0 is advanced to the next low byte.
- If the last address has not been examined (1264), this instruction produces a result equal to or greater than zero.
- If the result is equal to or greater than zero, examine the next memory location.

INSTRUCTION SET

The PDP-11 instruction set is presented in the following section. For ease of reference, the instructions are listed alphabetically.

SPECIAL SYMBOLS

You will find that a number of special symbols are used to describe certain features of individual instructions. The commonly used symbols are explained below.

SYMBOL	MEANING
MN	Maintenance Instruction
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
dst	Destination address
←	Becomes, or moves into. For example, (dst) ← (src) means that the source becomes the destination or that the source moves into the destination location.
(SP)+	Popped or removed from the hardware stack
−(SP)	Pushed or added to the hardware stack
Λ	Logical AND
v	Logical inclusive OR (either one or both)
V	Logical exclusive OR (either one, but not both)
~	Logical NOT
Reg or R	Register
B	Byte

NOTE

Condition code bits are considered to be cleared unless they are specifically listed as set.

INSTRUCTION SET

ADC
ADCB

Add Carry	SO	0055DD 1055DD

Operation: (dst)←(dst)+C

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if (dst) is 077777 and C = 1
C: set if (dst) is 177777 and C = 1

Description: Adds the contents of the C bit into 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	DO	06SSDD

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. 2's complement addition is performed.

ASH

Arithmetic Shift	DO	072RSS

Operation: R←R shifted arithmetically NN places to the right or left where NN = (src)

Conditions Codes:
N: set if result < 0
Z: set if result = 0
V: set if sign of register changed during shift
C: loaded from last bit shifted out of register

Description: The contents of the register are shifted right or left the number of times specified by the source operand. The shift count is taken as the low order 6 bits of the source operand. This number ranges from −32 to +31. Negative is a right shift and positive is a left shift.

NOTE
Standard for LSI-11/23; optionals for LSI-11/2 and LSI-11.

ASHC

Arithmetic Shift Combined DO 073RSS

Operation: R,Rv1←R,Rv1 The double word is shifted NN places to the right or left, where NN = (src)

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if sign bit changes during the shift
C: loaded with high order bit when left; loaded with low order bit when right shift (loaded with the last bit shifted out of the 32-bit operand)

Description: The contents of the register and the register OR-ed with 1 are treated as one 32-bit word. Rv1 (bits 0-15) and R (bits 16-31) are shifted right or left the number of times specified by the shift count. The shift count is taken as the low order 6 bits of the source operand. This number ranges from −32 to +31. Negative is a right shift and positive is a left shift.

When the register chosen is an odd number, the register and the register OR-ed with 1 are the same. In this case, the right shift becomes a rotate. The 16-bit word is rotated right the number of bits specified by the shift count.

NOTE
Standard for LSI-11/23; optionals for LSI-11/2 and LSI-11.

ASL
ASLB

Arithmetic Shift Left	SO	0063DD
	SO	1063DD

Operation: (dst) ←(dst) shifted one place to the left

Condition Codes:
N: set if high order bit of the 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 multiplicaton of the destination by 2 with overflow indication.

ASR
ASRB

Arithmetic Shift Right	SO	0062DD
	SO	1062DD

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 by 2.

BCC

Branch if Carry Clear	PC	103000

Operation: PC←PC+ (2 X offset) if C = 0

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Tests the state of the C bit and causes a branch if C is clear.

BCS

Branch if Carry Set   PC   1034000

Operation: PC←PC+ (2 X offset) if C = 1

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: 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.

BEQ

Branch if Equal   PC   001400

Operation: PC←PC+ (2 X offset) if Z = 1

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: 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.

BGE

Branch if Greater Than or Equal   PC   002000

Operation: PC←PC+ (2 X offset) if $N \veebar V = 0$

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: 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 of two positive numbers. BGE also causes a branch in a 0 result.

BGT

Branch if Greater Than   PC   003000

Operation: PC←PC+ (2 X offset) if Zv (N∀) = 0

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Causes a branch if the exclusive OR of the N and V bits is 1. Thus, BGT always branches following an operation that added two negative numbers, even if overflow occured. In particular, BGT always causes a branch if it follows a CMP instruction operating on a negative source and a positive destination (even if overflow occurred). Further, BGT never 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).

BHI

Branch if Higher   PC   101000

Operation: PC←PC + (2 X offset) if C = 0 and Z = 0

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Causes a branch if the previous operation causes 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.

BHIS

Branch if Higher Than the Same   PC   103000

Operation: PC←PC + (2 X offset) if C = 0

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Tests the state of the C bit and causes a branch if C is cleared.

BIC
BICB

Bit Clear   DO   04SSDD
14SSDD

Operation: (dst)← −(src)↑(dst)

Condition Codes:
N: set if high order bit of result set
Z: set if result = 0
V: cleared
C: not cleared

Description: Clears each bit in the destinaton that corresponds to a set bit in the source. The original contents of the destinaton are lost. The contents of the source are unaffected.

BIS
BISB

Bit Set   DO   05SSDD
15SSDD

Operation: (dst)←(src) v (dst)

Condition Codes:
N: set if high order bit of result set
Z: set if result = 0
V: cleared
C: not affected

Description: Performs 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
BITB

Bit Test	DO	03SSDD 13SSDD

Operation: (dst) (src)

Condition Codes:
N: set if high order bit of result set
Z: set if result = 0
V: cleared
C: not affected

Description: Performs 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.

BLE

Branch if Less Than or Equal To	PC	003400

Operation: PC← PC + (2 X offset) if Zv $(N \forall)=1$

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Causes a branch if the exclusive OR of the N and V bits is 1. Thus, BLE always branches following an operation that added two negative numbers, even if overflow occured. 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 postitive source and negative destination. BLE does not cause a branch if the result of the previous operation was 0 (without overflow).

BLO

Branch if Lower PC 1034000

Operation: PC← PC+ (2 X offset) if C = 1

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: 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.

BLOS

Branch if Lower or Same PC 101400

Operation: PC← PC + (2 X offset) if CvZ = 1

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: 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.

BLT

Branch if Less Than PC 002400

Operation: PC←PC + (2 X offset) if N∀ = 1

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Causes a branch if the exclusive OR of the N and V bits is 1. Thus, BLT always branches following 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 a 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).

BMI

Branch if Minus PC 100400

Operation: PC←PC + (2 X offset) if N = 1

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: 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

BNE

Branch if Not Equal PC 001000

Operation: PC←PC + (2 X offset) if Z = 0

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: 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.

BPL

Branch if Plus	PC	100000

Operation: PC←PC + (2 X offset) if N = 0

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Tests the state of the N bit and causes a branch if N is clear. BPL is the complementary operation of BMI.

BPT

Breakpoint Trap	PC	000003

Operation:
−(SP)←PS
−(SP)←PC
PC←(14)
PS←(16)

Condition Codes:
N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description: Performs a trap sequence with a trap vector address of 14. Used to call debugging aids. The user is cautioned against employing code 000003 in programs run under these debugging aids. No informaton is transmitted in the low byte.

BR

Branch	PC	0004000

Operation: PC←PC + (2 X offset)

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Provides a way of transferring program control within a range of −128 to +127 words with a 1-word instruction. An unconditional branch.

BVC

Branch if V Bit Clear	PC	102000

Operation: PC← PC + (2 X offset) if V = 0

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: 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.

BVS

Branch if V Bit Set	PC	102400

Operation: PC←PC + (2 X offset) if V = 1

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Tests the state of V bit (overflow) and causes a branch if the V bit is set. BVS is used to detect arithmetic overflow in the previous operation.

CCC

Clear all Condition Code Bits	CC	000257

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2, or 3, if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

Clear C	CC	000241	**CLC**

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2, 3, if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

CLN

Clear N	CC	000250

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2, or 3, if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

CLR
CLRB

Clear	SO	0050DD 1050DD

Operation: (dst)←0

Condition Codes:
N: cleared
Z: set
V: cleared
C: cleared

Description: Contents of specified destination are replaced with 0s.

NOTE

As a performance optimization, on the LSI-11/23 the last bus cycle of a CLR (or CLRB) is a DATO (or DATOB). LSI-11 and LSI-11/2 processors performed a DATIO cycle for the last bus cycle as a "don't care" for hardware minimization.

CLV

Clear V	CC	000242

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2, or 3, if bit 4 is a 1. Clears corresponding bit 4 = 0.

CLZ

Clear Z	CC	000244

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2, or 3, if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

CMP
CMPB

Compare	DO	02SSDD 12SSDD

Operation: (src) - (dst) [in detail (src) + (dst) +1]

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if there is arithmetic overflow; i.e., operands of opposite signs and the sign of the destination is the same as the sign of the result
C: cleared if there is a carry from the most significant bit of the result

Description: Compares the source and destinaton 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.

COM
COMB

Complement	SO	0051DD 1051DD

Operation: (dst)←n(dst)

Condition Codes:
N: set if most significant bit of result = 0
Z: set if result = 0
V: cleared
X: 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).

DEC
DECB

Decrement	SO	0053DD 1053DD

Operation: (dst)← (dst) − 1

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if (dst) was 100000
C: not affected

Description: Subtract 1 from the contents of the destination.

DIV

Divide	DO	071RSS

Operation: R,Rv1←R,Rv1/(src)

Condition Codes:
N: set if quotient < 0
Z: set if quotient = 0
V: set if source = 0 or if the absolute value of the register is larger than the absolute value of instruction is the source. (In this case the would exceed 15 instruction is aborted because the quotient would exceed 15 bits.)
C: set if divide by 0 attempted

Description:	The 32-bit 2's complement integer in R and Rv1 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.

NOTE
Standard for LSI-11/23; optionals
for LSI-11/2 and LSI-11.

EMT

Emulator Trap PC 104000

Operation:
−(SP)←PS
−(SP)←PC
PC←(30)
PS←(32)

Condition Codes:
N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description: All operation codes from 104000 to 104377 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; the new process status (PS) is taken from the word at address 32.

Caution: EMT is used frequently by DIGITAL system software and is therefore not recommended for general use.

HALT

Halt MS 000000

Operation:

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Causes program execution to cease and enters console ODT (if memory management is present, program execution ceases only if in kernel mode; a trap to location 10 occurs if in user mode). Additionally if jumper - on the KDF11 module is inserted, a trap to 10 will occur unconditionally.

INC
INCB

Increment	SO	0052DD 1052DD
Operation:	(dst)← (dst) + 1	
Condition Codes:	N: set if result < 0 Z: set if result = 0 V: set if dst was 077777 C: not affected	
Description:	Adds 1 to the contents of the destination.	

IOT

I/O trap	PC	000004

Operation:	−(SP)←PS
	−(SP)←PC
	PC←(20)
	PS←(22)	

Condition Codes:	N: loaded from trap vector Z: loaded from trap vector V: loaded from trap vector C: loaded from trap vector	
Description:	Performs a trap sequence with a trap vector address of 20. 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.	

JMP

Jump	PC	0001DD
Operation:	PC←(dst)	
Condition Codes:	N: unaffected Z: unaffected V: unaffected C: unaffected	

Description: JMP provides more flexible program branching than provided with the branch instruction. It is not limited to $+177_8$ and -200_8 words as are branch instructions. JMP does generate a second word, 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 will cause an illegal instruction condition and a trap to location 4. (Program control cannot be transferred to a register.) Register deferred mode is legal and will cause 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.

JSR

Jump to Subroutine PC 004RDD

Operation: (tmp)←(dst)(tmp is an internal processor register) - (SP)←reg (push reg contents onto processor stack) reg ← PC PC holds location following JSR; this address now put in reg PC ← (tmp) PC now points to subroutine address

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: 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 re-entrant manner on the processor stack, execution of a subroutine may be interrupted, and the same subroutine re-entered 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 subroutine call suitable for subroutine calls that transmit parameters. JSR, PC saves the use of an extra register.

In both JSR and JMP the address is used to load the program counter, R7. Thus, for example, a JSR is destination mode 1 for general register R1 (where (R1) = 100) will access a subroutine at location 100. This is effectively one level less of deferral than operate instructions such as add.

A JSR with mode 0 will result in an illegal instruction and a trap through the trap vector address 4.

MARK

Mark	PC	0064NN

Operation:
SP←PC+2Xnn
PC←R5
R5←(SP)+ nn = number of parameters

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Used as part of the standard subroutine return convention. MARK facilitates the stack clean-up procedures involved in subroutine exit. Assembler format is: MARK N

MFPD
MFPI

Move from Previous Data Space	MS	0065SS
Move from Previous Instruction Space		0065SS

Operation:	(temp)←(src) - (SP)←(temp)
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 stack from an address in previous space. The source address is computed using the current registers and memory map. Since data space does not exist in the KDF11, MFPD executes the same as a MFPI. (LSI-11/23 only).

MFPS

Move Byte from Processor Status Word	MS	1067DD

Operation:	(dst)←PS dst lower 8 bits
Condition Codes:	N: set if PS bit 7 = 1 Z: set if PS <0:7> = 0 V: cleared C: not affected
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 upper byte of the register. The destination operand is treated as a byte address. The KDF11 implements the PS address, 777776, which can be used as another method of accessing the PS. This method can be used on all PDP-11s except LSI-11 and LSI-11/2 processors.

MOV
MOVB

Move	DO	01SSDD 11SSDD

Operation:	(dst)←(src)

Condition Codes:	N: set if (src) < 0 Z: set if (src) = 0 V: cleared C: not affected
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) (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.

NOTE

As a performance optimization, on the LSI-11/23 the last bus cycle of a MOV (or MOVB) is a DATO (or DATOB). LSI-11 and LSI-11/2 processors performed a DATIO cycle for MOVB as a "don't care" for hardware minimization.

MTPD
MTPI

Move to Previous Data Space	MS	1066SS
Move to Previous Instruction Space		0066SS

Operation:	(temp)←(SP) + (dst)←(temp)
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 (bits 15, 14) and stores that word

into an address in previous space PS (bits 13, 12). The destination address is computed using the current registers and memory map.

Since data space does not exist in the KDF11, MTPD executes the same as MTPI. (LSI-11/23 only).

NOTE
As a performance optimization, on the LSI-11/23 the lost bus cycle of a MTPD and MTPI is a DATO. This instruction was not implemented on LSI-11 and LSI-11/2 processors.

MTPS

Move Byte to Processor Status Word	MS	1064SS

Operation: PS←(src)

Condition Codes:
N: set according to effective src operand 0-3
Z: same
C: same

Description: The 8-bits of the effective operand replace the current low byte contents of the PS, 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 seen with this instruction in either kernel or user mode. The src operand remains unchanged.

The KDF11 implements the PS address, 777776, which can be used as another method of accessing the PS. This method can be used on all PDP-11s except previous LSI-11 processors.

MUL

Multiply	DO	070RSS

Operation: R,Rv1←RX(src)   even Reg is hi-order product
odd Reg is lo-order product

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. Assembler syntax is:

MUL S,R. (Note that the actual destination is R, Rv1, which reduces to just R when R is odd.)

NOTE

Standard for LSI-11/23; optionals for LSI-11/2 and LSI-11.

NEG
NEGB

Negate SO 0054DD
1054DD

Operation: (dst)←(dst)

Condition Codes:
N: set if result < 0
Z: set if result = 0
V: set if result = 100000
C: cleared if result = 0

Description: Replaces the contents of the destination address by its 2's complement. Note that 100000 is replaced by itself.

RESET

Return External Bus MS 000005

Operation: PC(SP)
PS(SP)

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Causes bus signal BINITL to be asserted for 10 μs and then unasserted for 90 μs. Used to initialize I/O devices attached to the bus. In addition memory management status registers SR0 and SR3 are cleared.

ROL
ROLB

Rotate Left	SO	0061DD 1061DD

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 exclusive OR of the N bit and C bit (as set by the completion of the rotate operation)
C: loaded with the high order bit of the destination

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.

ROR
RORB

Rotate Right	SO	0060DD

Operation: (dst)←(dst) rotate right one place

Condition Codes:
N: set if high order bit of the result is set
Z: set if all bits of result are 0
V: loaded with the exclusive OR of the N bit and the C bit as set by ROR
C: loaded with the low order bit of the destination

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 destintation.

RTI

Return from Interrupt	MS	000002

Operation: PC←(SP) + PS←(SP) +

Condition Codes:
N: loaded from processor stack
Z: loaded from processor stack
V: loaded from processor stack
C: loaded from processor 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.

RTS

Return from Subroutine PC 00020R

Operation: PC←(reg) (reg)←SP+

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: unaffected

Description: Loads contents of register into 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.

RTT

Return from Interrupt MS 000006

Operation: PC←(SP) + PS←(SP)+

Condition Codes:
N: loaded from processor stack
Z: loaded from processor stack
V: loaded from processor stack
C: loaded from processor stack

Description: Used to exit from a trace trap (T bit) service routine and executes the same as the RTT instruction 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.

SBC
SBCB

Subtract Carry	SO	0056DD 1056DD
Operation:	(dst)←(dst)-C	
Condition Codes:	N: set if result < 0 Z: set if result = 0 V: set if (dst) = 100000 and C = 1 C: cleared if (dst) = 0 and C = 1	
Description:	Subtracts 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.	

SCC

Set all Cs	CC	000277

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2, or 3, if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

SEC

Set C	CC	000261

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2, or 3, if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

SEN

Set N	CC	000262

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator, i.e., sets the bit specified by bit 0, 1, 2, or 3, if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

SEV

Set V	CC	000263

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2, or 3, if bit 4 is a 1. Clears corresponding bits if bit 4 = 0.

SEZ

Set Z	CC	000264

Sets and clears condition code bits. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified according to the sense of bit 4, the set/clear bit of the operator; i.e., sets the bit specified by bit 0, 1, 2, 3, if bit 4 is a 1. Clears corresponding bits if 4 = 0.

SOB

Subtract One and Branch if not Equal to 0	PC	077R00 plus offset

Operation:	R←R−1 if this result does not = 0 then PC←PC−(2 × offset)
Condition Codes:	N: unaffected Z: unaffected V: unaffected C: 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. Assembler syntax is:

SOB R,A

where A is the address to which transfer is to be made if the decremented R is not equal to 0. Note that the SOB instruction cannot be used to transfer control in the forward direction.

SUB

Subtract	DO	16SSDD

Operation: (dst)←(dst)-(src)

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 carry from the most significant bit of the result

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, when set, indicates a borrow.

SWAB

Swap Byte	SO	0003DD

Operation: Byte 1/Byte 0
Byte 0/Byte 1

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 high order byte and low order byte of the destination (which must be a word address).

SXT

Sign Extend	SO	0067DD

Operation:	(dst)←0 if N 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, then a −1 is placed in the destination operand; if N bit is clear, then 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.

NOTE

As a performance optimization, on the LSI-11/23 the last bus cycle of a SXT is a DATO. LSI-11/23 and LSI-11/2 processors performed a DATIO cycle for the last bus cycle as a "don't care" for hardware minimization.

TRAP

Trap	PC	104400 to 104777

Operation:	−(SP)←PS
	−(SP)←PC
	PC←(34)
	PS←(36)

Condition Codes:	N: loaded from trap vector Z: loaded from trap vector V: loaded from trap vector C: loaded from trap vector

Description: Operation codes from 104400 to 104777 are TRAP instructions. TRAPs and EMTs are identical in operation, except that the trap vector for TRAP is at address 34.

NOTE

Since DIGITAL software makes frequent use of EMT, the TRAP instruction is recommended for general use.

TST
TSTB

Test	SO	0047DD 1057DD

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.

WAIT

Wait for Interrupt	MS	000001

Operation:

Condition Codes:
N: unaffected
Z: unaffected
V: unaffected
C: 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 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.

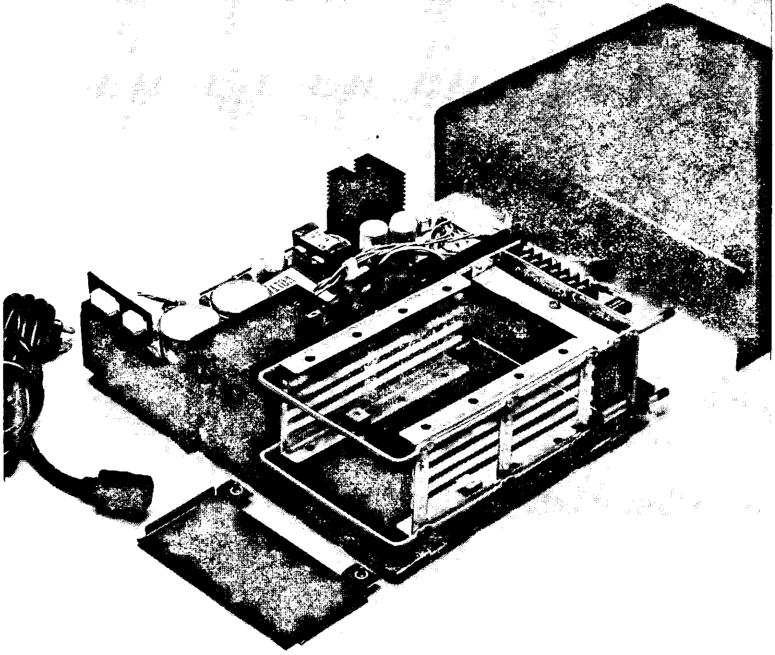
XOR

Exclusive OR	DO	074RDD

Operation: (dst)←Rv(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. Contents of register are unaffected. Assembler format is XOR R, D.



CHAPTER 8
LSI-11 BUS

The LSI-11 Bus, also called the sub-UNIBUS and the Q-Bus, is the low end member of DIGITAL's bus family.

The LSI-11 Bus consists of 36 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:

- 18 Data/address lines — BDAL<17:00>
- 6 Data Transfer control lines — BBS7, BDIN, BDOUT, BRPLY, BSYNC, BWTBT
- 3 Direct memory access control lines — BDMG, BDMR, BSACK
- 6 Interrupt control lines — BEVNT, BIAK, BIRQ4, BIRQ5, BIRQ6, BIRQ7
- 5 System control lines — BDCOK, BHALT, BINIT, BPOK, BREF

Most LSI-11 bus signals are bidirectional and use terminations for a negated (high) signal level. Devices connect to these lines via high-impedance 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 (BIACK) 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.

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., who 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 timeout 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 timeout error occurs must be longer than the reply time of the slowest peripheral or memory device on the bus. The signals and pin assignments are shown in the table that follows. The pin nomenclature is for reference and is only required when examining DIGITAL modules and circuit schematics.

Number of Pins	**Functional Category**	**DIGITAL's Nomenclature (Pin Name)**					
18	Data/Address	BDAL0	BDAL1	BDAL2	BDAL15	BDAL16	BDAL17
		AU2	AV2	BE2	BV2	AC1	AD1
6	Data	BDOUT	BRPLY	BDIN	BSYNC	BWTBT	BBS7
	Control	AE2	AF2	AH2	AJ2	AD2	AP2
6	Interrupt	BIRQ7	BIRQ6	BIRQ5	BIRQ4	BIAKO	BIAKI
	Control	BPI	AB1	AA1	AL2	AN3	AM2
4	DMA	BDMR	BDGO	BDMG1	BSACK		
	Control	AN1	AS2	AR2	BN1		
6	System	BHALT	BREF	BDCOK	BPOK	BEVNT	BINIT
	Control	AP1	AR1	BA1	BB1	BR1	AT2

Number of Pins	**Functional Category**	**DIGITAL's Nomenclature (Pin Name)**
3	+5 Vdc	AA2, BA2, BV1
2	+12 Vdc	AD2, BD2
2	−12 Vdc	AB2, BB2
2	+12B (battery)	AS1, BS1
1	+5B (battery)	AV1, (AE1, AS1, alternates)
8	GND	AC2, AJ1, AM1, AT1, BC2, BJ1, BM1, BT1
8	S SPARES	AE1, AF1, AH1, BC1, BD1, BE1, BF1, BH1
4	M SPARES	AK1, AL1, BK1, BL1
2	P SPARES	AU1, BU1

DATA TRANSFER BUS CYCLES

Data transfer bus cycles are as follows.

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

These bus cycles, executed by bus master devices, transfer 16-bit words or 8-bit bytes to or from slave devices. The following bus signals are used in a data transfer operation.

Mnemonic	Description	Function
BDAL<17:00> L	18 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.
BSYNC L	Bus Cycle Control	Strobe signals
BDIN L	Data input indicator	
BDOUT L	Data output indicator	
BRPLY L	Slave's acknowledgement of bus cycle	
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 three basic types: DATI, DATO(B), and DATIO(B). 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/deskew time. During the address setup and deskew time the bus master does the following:

- asserts BDAL<17:00> 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) 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 five high-order address bits BDAL<17:13> and instead decode BBS7 L along with the thirteen low-order address bits. An active BWTBT L signal indicates that a DATO(B) operation follows, while an inactive BWTBT L indicates a DATI or DATIO(B) operation.

The address hold/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<17:00> 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<17:13> 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 is a read operation. During DATI data is input to the bus master. Data consists 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 in the following ways:

- asserts BRPLY L after receiving BDIN L and 125 ns (maximum) before BDAL bus driver data bits are valid
- asserts BDAL<17:00> L with the addressed data and error information

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 transmittig parity errors to the master.
- Negates BDIN L 150 ns (minimum) to 2 microseconds (maximum) after BRPLY L goes active.

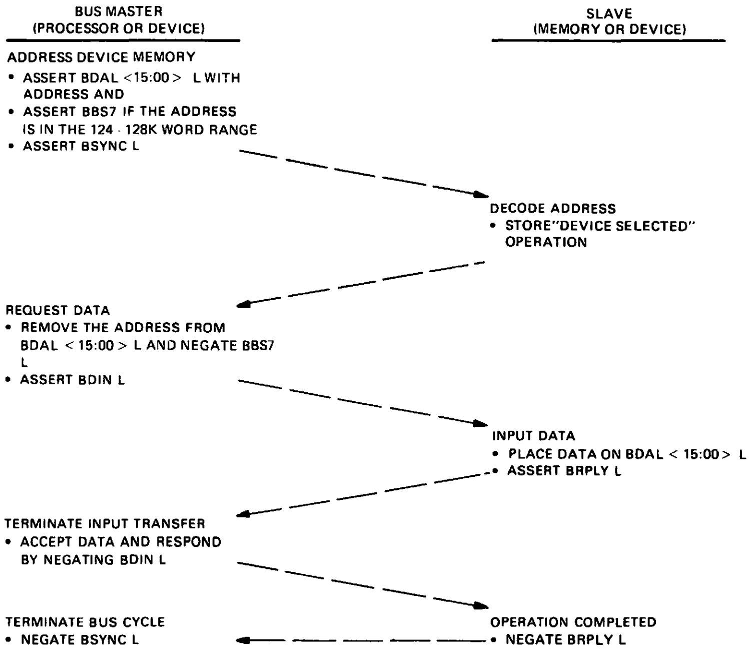
The slave device responds to BDIN L negation by negating BRPLY L and removing read data from BDAL bus drivers. BRPLY L must be negated 100 ns (maximum) prior to removal of read data. The bus master responds to the negated BRPLY L by negating BSYNC L.

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

NOTE

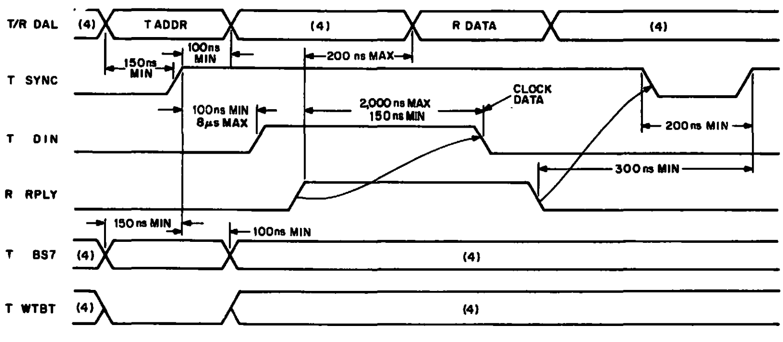
Continuous assertion of BSYNC L retains control of the bus by the bus master, and the previously addressed slave device remains selected. This is done for DATIO(B) bus cycles where DATO or DATOB follows a DATI without BSYNC L negation and a second device addressing operation. Also, a slow slave device can hold off data transfers to itself by keeping BRPLY L asserted, which will cause the master to keep BSYNC L asserted.



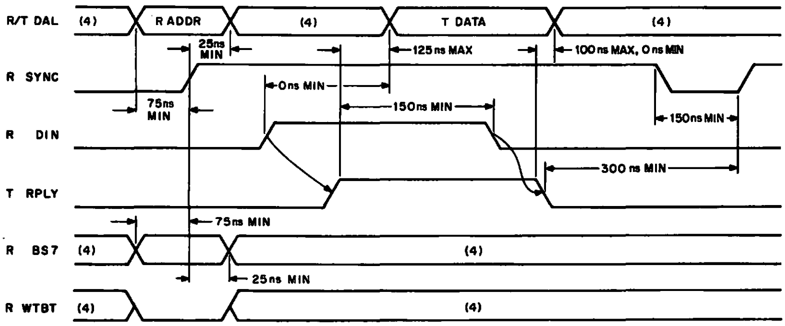
BUS MASTER (PROCESSOR OR DEVICE)
SLAVE (MEMORY OR DEVICE)
ADDRESS DEVICE MEMORY
• ASSERT BDAL <15:00> L WITH ADDRESS AND
• ASSERT BBS7 IF THE ADDRESS IS IN THE 124 - 128K WORD RANGE
• ASSERT BSYNC L
DECODE ADDRESS
• STORE "DEVICE SELECTED" OPERATION
REQUEST DATA
• REMOVE THE ADDRESS FROM BDAL < 15:00 > L AND NEGATE BBS7 L
• ASSERT BDIN L
INPUT DATA
• PLACE DATA ON BDAL < 15:00 > L
• ASSERT BRPLY L
TERMINATE INPUT TRANSFER
• ACCEPT DATA AND RESPOND BY NEGATING BDIN L
TERMINATE BUS CYCLE
• NEGATE BSYNC L
OPERATION COMPLETED
• NEGATE BRPLY L

DATI Bus Cycle

DATO(B) — DATO(B) is a write operation. Data is transfered 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 had been asserted by the bus master, or immediately following an input transfer part of a DATIO(B) bus cycle.



TIMING AT MASTER DEVICE

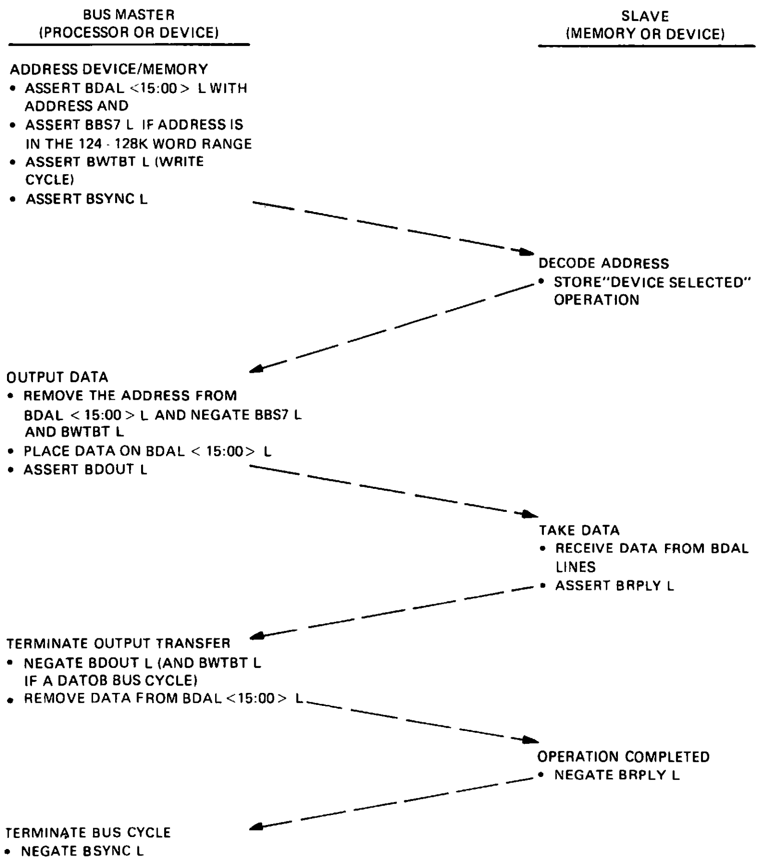


TIMING AT SLAVE DEVICE

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.

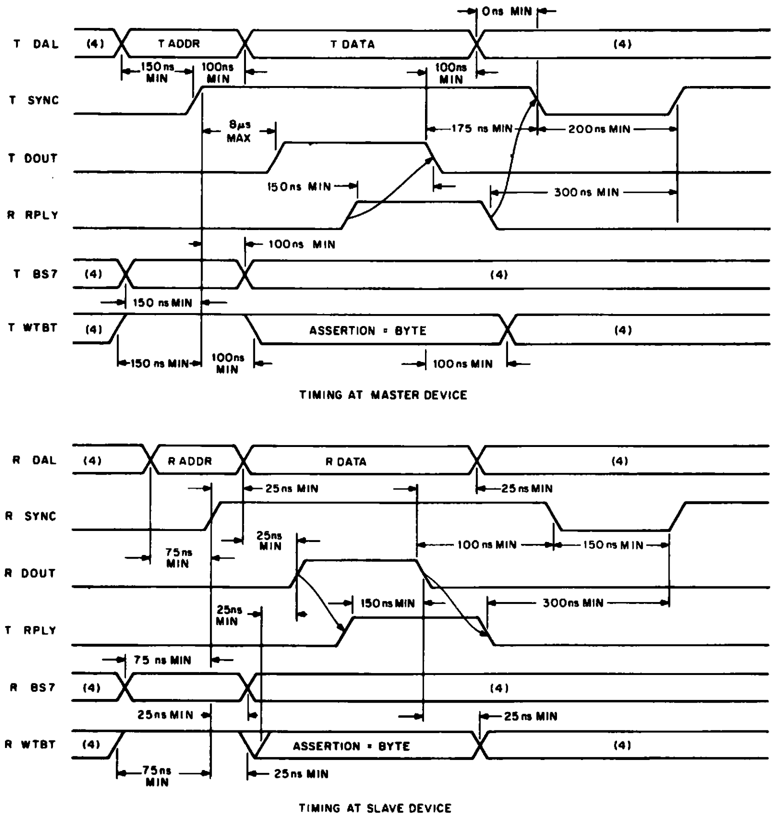
DATI Bus Cycle Timing


BUS MASTER
(PROCESSOR OR DEVICE)
SLAVE
(MEMORY OR DEVICE)
ADDRESS DEVICE/MEMORY
• ASSERT BDAL <15:00> L WITH ADDRESS AND
• ASSERT BBS7 L IF ADDRESS IS IN THE 124 - 128K WORD RANGE
• ASSERT BWTBT L (WRITE CYCLE)
• ASSERT BSYNC L
DECODE ADDRESS
• STORE"DEVICE SELECTED" OPERATION
OUTPUT DATA
• REMOVE THE ADDRESS FROM BDAL < 15:00 > L AND NEGATE BBS7 L AND BWTBT L
• PLACE DATA ON BDAL < 15:00> L
• ASSERT BDOUT L
TAKE DATA
• RECEIVE DATA FROM BDAL LINES
• ASSERT BRPLY L
TERMINATE OUTPUT TRANSFER
• NEGATE BDOUT L (AND BWTBT L IF A DATOB BUS CYCLE)
• REMOVE DATA FROM BDAL <15:00> L
OPERATION COMPLETED
• NEGATE BRPLY L
TERMINATE BUS CYCLE
• NEGATE BSYNC L

DATO or DATOB 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<16:00> L at least 100 ns after BSYNC L is asserted if the transfer is a word transfer. If it is 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. During a byte transfer, BDAL 00 L selects the high or low byte.



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

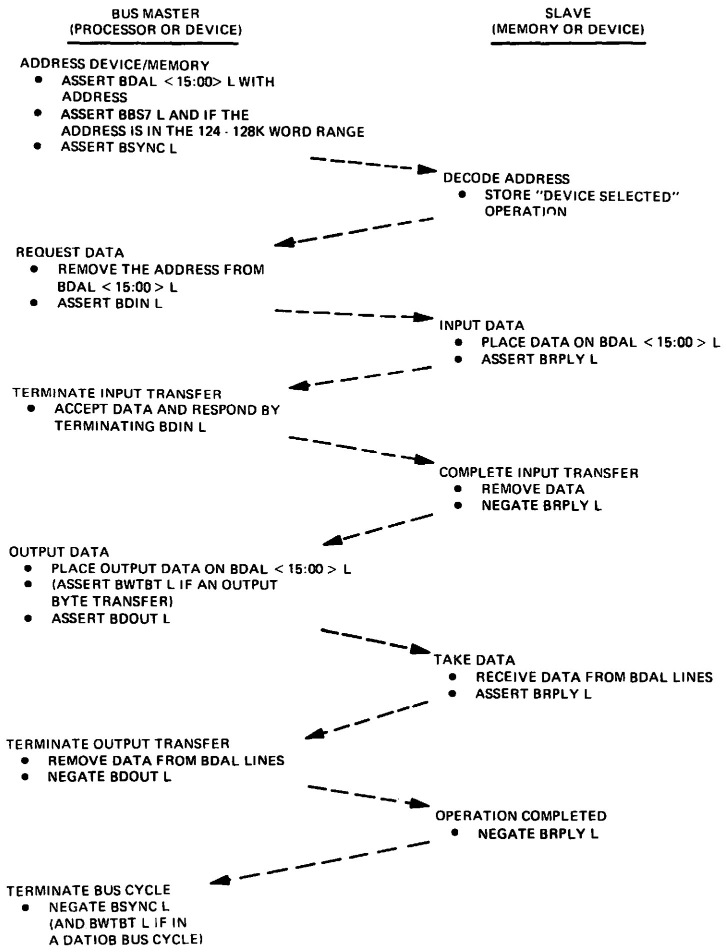
DATO or DATOB Bus Cycle Timing

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 BDAL 16 L at this time will force a parity error to be written into memory if the memory is a parity-type memory. BDAL 17 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 timeout. This completes the data setup and deskew time.

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 sense BDOUT L negation. The data is 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.

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. 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, which is the same as described for DATO(B).



BUS MASTER
(PROCESSOR OR DEVICE)

SLAVE
(MEMORY OR DEVICE)

ADDRESS DEVICE/MEMORY
- ASSERT BDAL <15:00> L WITH ADDRESS
- ASSERT BBS7 L AND IF THE ADDRESS IS IN THE 124 - 128K WORD RANGE
- ASSERT BSYNC L

DECODE ADDRESS
- STORE "DEVICE SELECTED" OPERATION

REQUEST DATA
- REMOVE THE ADDRESS FROM BDAL < 15:00 > L
- ASSERT BDIN L

INPUT DATA
- PLACE DATA ON BDAL < 15:00 > L
- ASSERT BRPLY L

TERMINATE INPUT TRANSFER
- ACCEPT DATA AND RESPOND BY TERMINATING BDIN L

COMPLETE INPUT TRANSFER
- REMOVE DATA
- NEGATE BRPLY L

OUTPUT DATA
- PLACE OUTPUT DATA ON BDAL < 15:00 > L
- (ASSERT BWTBT L IF AN OUTPUT BYTE TRANSFER)
- ASSERT BDOUT L

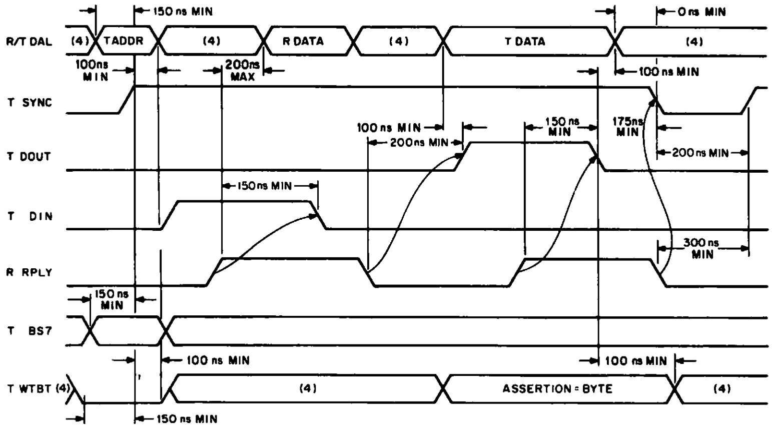
TAKE DATA
- RECEIVE DATA FROM BDAL LINES
- ASSERT BRPLY L

TERMINATE OUTPUT TRANSFER
- REMOVE DATA FROM BDAL LINES
- NEGATE BDOUT L

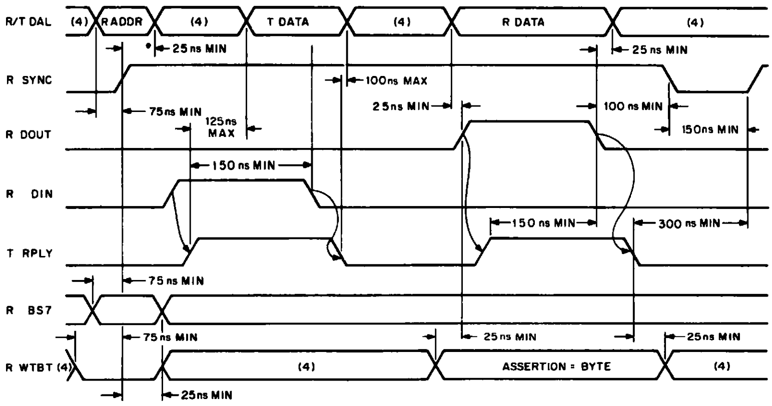
OPERATION COMPLETED
- NEGATE BRPLY L

TERMINATE BUS CYCLE
- NEGATE BSYNC L (AND BWTBT L IF IN A DATIOB BUS CYCLE)

DATIO or DATIOB Bus Cycle



TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

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

DATIO OR DATIOB 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 only needs to know 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 the processor. 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

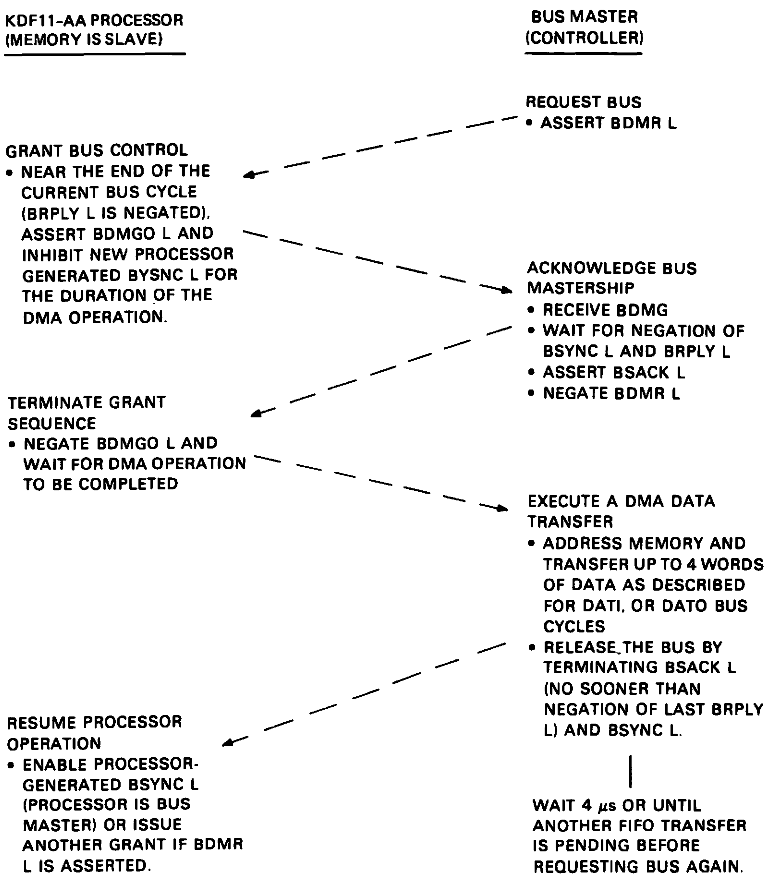
DMA Protocol

A DMA transaction can be divided into three phases:

- the bus mastership acquisition phase
- the data transfer phase
- the 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. 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 no device responds to the DMA grant the processor will clear the grant and rearbitrate the bus. If BDMR L is continously asserted, the bus will be hung (the grant signal will keep passing down the bus, be cleared after no BSACK L occurs, and be driven again after the bus is rearbitrated).

During the data transfer phase, the DMA device continues asserting BSACK L. The actual data transfer is performed as described earlier.



KDF11-AA PROCESSOR (MEMORY IS SLAVE)

BUS MASTER (CONTROLLER)

REQUEST BUS
- ASSERT BDMR L

GRANT BUS CONTROL
- NEAR THE END OF THE CURRENT BUS CYCLE (BRPLY L IS NEGATED), ASSERT BDMGO L AND INHIBIT NEW PROCESSOR GENERATED BYSNC L FOR THE DURATION OF THE DMA OPERATION.

ACKNOWLEDGE BUS MASTERSHIP
- RECEIVE BDMG
- WAIT FOR NEGATION OF BSYNC L AND BRPLY L
- ASSERT BSACK L
- NEGATE BDMR L

TERMINATE GRANT SEQUENCE
- NEGATE BDMGO L AND WAIT FOR DMA OPERATION TO BE COMPLETED

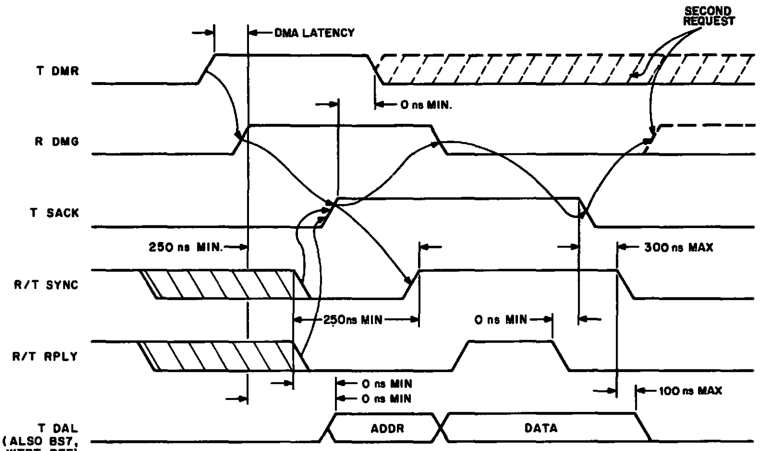
EXECUTE A DMA DATA TRANSFER
- ADDRESS MEMORY AND TRANSFER UP TO 4 WORDS OF DATA AS DESCRIBED FOR DATI, OR DATO BUS CYCLES
- RELEASE THE BUS BY TERMINATING BSACK L (NO SOONER THAN NEGATION OF LAST BRPLY L) AND BSYNC L.

WAIT 4 μs OR UNTIL ANOTHER FIFO TRANSFER IS PENDING BEFORE REQUESTING BUS AGAIN.

RESUME PROCESSOR OPERATION
- ENABLE PROCESSOR-GENERATED BSYNC L (PROCESSOR IS BUS MASTER) OR ISSUE ANOTHER GRANT IF BDMR L IS ASSERTED.

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).



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.

DMA Request/Grant Timing

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.

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 300 ns (maximum) before negating BSYNC L.

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. Interrupt operations can also originate from within the processor. These interrupts are called *traps*. Traps are caused by programming errors, hardware errors, special instructions and maintenance features.

The LSI-11 bus signals that are used in interrupt transactions are the following:

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<15:00> L	Data/address lines
BDIN L	Data input strobe
BRPLY L	Reply

LSI-11 and LSI-11/2 processors recognize interrupt requests on BIRQ4 only. All present and future interfaces will operate compatibly with these processors on a single level interrupt basis.

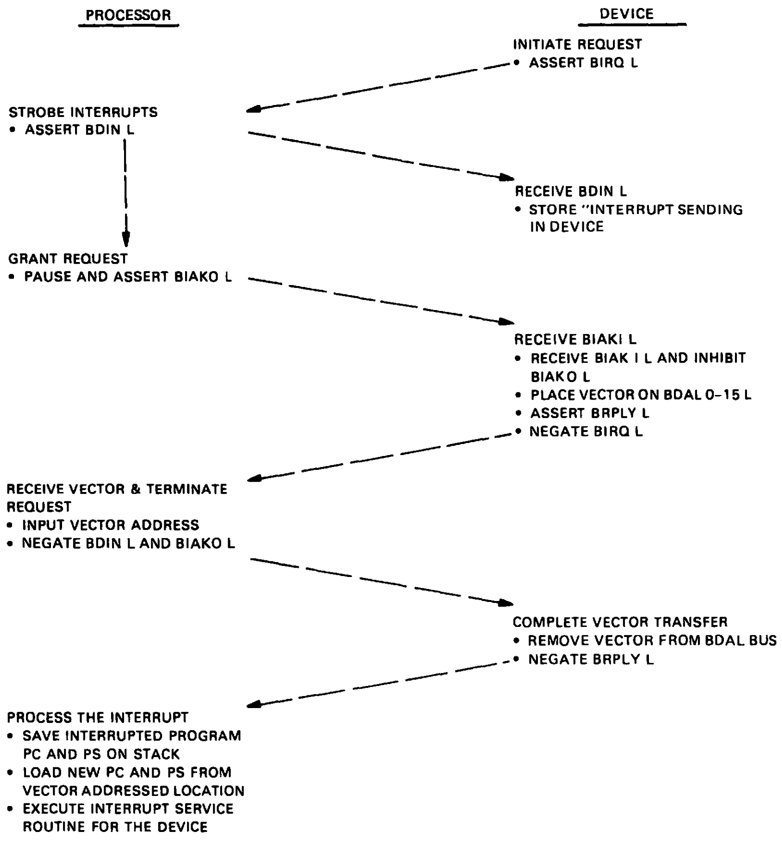
Device Priority

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.

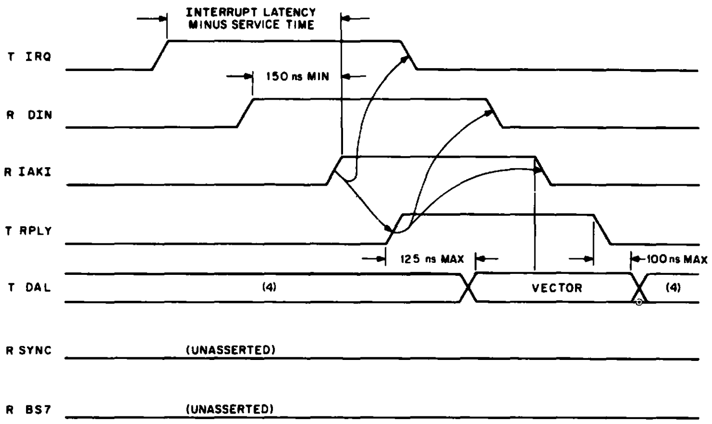
Interrupt Protocol

Interrupt protocol on the LSI-11/23 has three phases: interrupt request phase, interrupt acknowledge and priority arbitration phase, and interrupt vector transfer phase.



PROCESSOR
DEVICE
INITIATE REQUEST
• ASSERT BIRQ L
STROBE INTERRUPTS
• ASSERT BDIN L
RECEIVE BDIN L
• STORE "INTERRUPT SENDING IN DEVICE
GRANT REQUEST
• PAUSE AND ASSERT BIAKO L
RECEIVE BIAKI L
• RECEIVE BIAK I L AND INHIBIT BIAKO L
• PLACE VECTOR ON BDAL 0-15 L
• ASSERT BRPLY L
• NEGATE BIRQ L
RECEIVE VECTOR & TERMINATE REQUEST
• INPUT VECTOR ADDRESS
• NEGATE BDIN L AND BIAKO L
COMPLETE VECTOR TRANSFER
• REMOVE VECTOR FROM BDAL BUS
• NEGATE BRPLY L
PROCESS THE INTERRUPT
• SAVE INTERRUPTED PROGRAM PC AND PS ON STACK
• LOAD NEW PC AND PS FROM VECTOR ADDRESSED LOCATION
• EXECUTE INTERRUPT SERVICE ROUTINE FOR THE DEVICE

Interrupt Request/Acknowledge Sequence



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

Interrupt Protocol Timing

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 item 2 of the arbitration discussion involving the 4-level scheme (below) 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

The interrupt request line remains asserted until the request is acknowledged.

During the interrupt acknowledge and priority arbitration phase the processor LSI-11/23 will acknowledge interrupts under the following conditions:

1. The device interrupt priority is higher than the current PS<07: 05>.
2. The processor has completed instruction execution and no additional bus cycles are pending.

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 monitor the level 6 request. This protocol has been optimized for levels 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 the figure illustrating interrupt protocol timing.

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

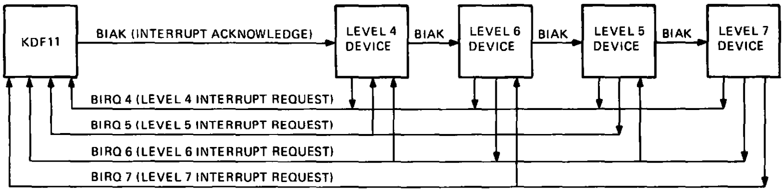
Propagation delay from BIAKI L to BIAKO L must be greater than 500 ns per LSI-11 bus slot.

The device must assert BRPLY L within 10 microseconds (maximum) after the processor asserts BIAKI L.

LSI-11/23 4-Level Interrupt Configurations

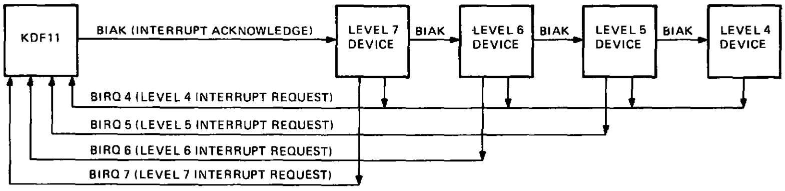
If you have high-speed peripherals and desire better software performance you can use the 4-level interrupt scheme. Both position-independent and position-dependent configurations can be used with the 4-level interrupt scheme.

The position-independent configuration is illustrated in the accompanying figure. 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 reqardless 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 properly function.


KDF11
BIAK (INTERRUPT ACKNOWLEDGE)
LEVEL 4 DEVICE
BIAK
LEVEL 6 DEVICE
BIAK
LEVEL 5 DEVICE
BIAK
LEVEL 7 DEVICE
BIRQ 4 (LEVEL 4 INTERRUPT REQUEST)
BIRQ 5 (LEVEL 5 INTERRUPT REQUEST)
BIRQ 6 (LEVEL 6 INTERRUPT REQUEST)
BIRQ 7 (LEVEL 7 INTERRUPT REQUEST)

Position-Independent Configuration

The position-dependent configuration is illustrated in the accompanying figure. 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 only has to assert 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.


KDF11
BIAK (INTERRUPT ACKNOWLEDGE)
LEVEL 7 DEVICE
BIAK
LEVEL 6 DEVICE
BIAK
LEVEL 5 DEVICE
BIAK
LEVEL 4 DEVICE
BIRQ 4 (LEVEL 4 INTERRUPT REQUEST)
BIRQ 5 (LEVEL 5 INTERRUPT REQUEST)
BIRQ 6 (LEVEL 6 INTERRUPT REQUEST)
BIRQ 7 (LEVEL 7 INTERRUPT REQUEST)

Position-Dependent Configuration

CONTROL FUNCTIONS

The following LSI-11 bus signals provide control functions.

BREF L	Memory refresh
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 simultaneously addressed. 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.

Halt
Assertion of BHALT L 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 or as part of a power-up sequence. 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) and are defined as follows.

BDCOK H
The assertion of this line indicates that dc power has been stable for at least 3 ms. Once asserted this line remains asserted until the power fails. The negation of this line is the first event in the power-fail sequence. It indicates that only 5 microseconds of dc power reserve remains. Once BDCOK H is negated it must remain in this state for at least 1 microsecond before being asserted again.

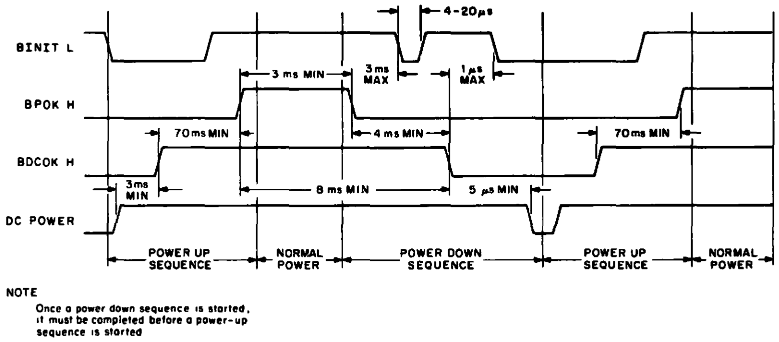
BPOK H
The assertion of this line indicates that there is at least an 8 ms reserve of dc power and that BDCOK H has been asserted for at least 70 ms. Once BPOK H has been asserted, it must remain asserted for at least 3 ms. The negation of this line indicates that power is failing and that only 4 ms 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. The LSI-11/23 has four power-up jumper options.

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. The processor traps to location 24_8> Location 24_8contains the PC that points to the power-down routine. 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.



Power-Up/Power-Down Timing

BUS ELECTRICAL CHARACTERISTICS

This paragraph contains information about the electrical characteristics of the LSI-11 bus.

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

AC 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 Load Definition
DC loads are defined as maximum current allowed with a signal line driver asserted or unasserted. A unit load is defined as 105 μA in the unasserted state.

120 Ohm LSI-11 Bus
The electrial 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. Insofar as bus drivers, receivers and wiring connected to the bus have finite resistance and nonzero reactance, the transmission line impedance becomes non-uniform, and thus 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: 0.7 V maximum

Output high leakage current when connected to 3.8 Vdc: 25 uA (even if no power is applied to them, 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): 1.3 V

Input high voltage (minimum): 1.7 V

Maximum input current when connected to 3.8 Vdc: 80 μA even if no power is applied to them.

These specifications must be met at worst-case supply voltage, temperature, and output signal conditions.

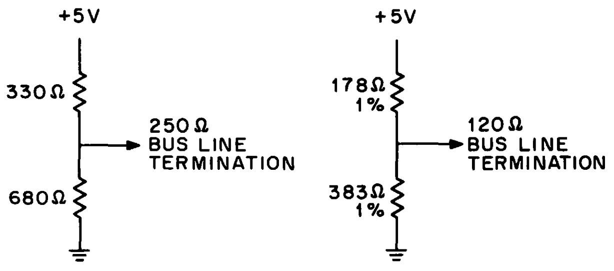
AC Specifications

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. This is to be done as a voltage divider with its Thevenin equivalent equal to 120 ohms and 3.4 V nominal. This type of termination is provided by a REV11-A refresh/boot/terminator, or the BDV11-AA.



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): z = 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). Both of these two terminators must be physically resident within the same backplane.

Bus Interconnecting Wiring
This paragraph contains the electrical characteristics of the bus interface.

Backplane Wiring — The wiring that interconnects 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 signal path, as measured between near-end terminator and 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 near-end terminator and 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 interconnects the bus connector slots within one contiguous backplane is part of the overall bus transmission line. Due to implementation constraints, the nominal characteristic impendance 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%; maximum ripple: 100 mV pp. +12 Vdc must be regulated to ±3%; maximum ripple: 200 mV pp.

+5 Vdc — 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 include:

1. Power consumption — +5 Vdc and +12 Vdc current requirements.
2. 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.
3. 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 105 microamperes (nominal).

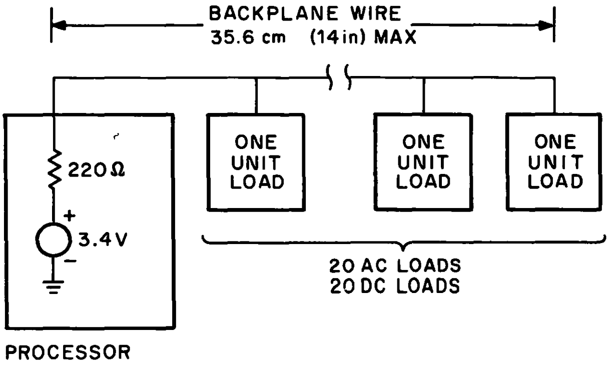
Power consumption, ac loading, and dc loading specifications for each module are included in the *Microcomputer Interface* handbook.

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

- The bus can accommodate modules that have up to 20 ac loads (total) before an additional termination is required. The processor has on-board termination for one end of the bus. If more than 20 ac loads are included, the other end of the bus must be terminated with 120 ohms.
- A single backplane terminated bus can accommodate modules of up to 35 ac loads (total).
- 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.

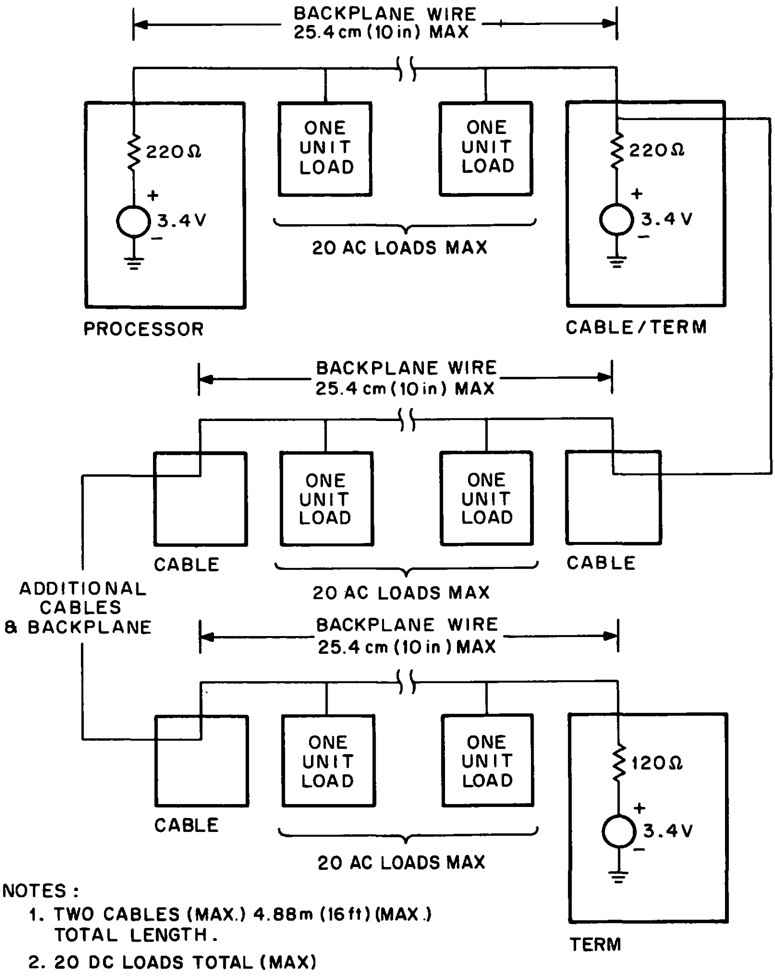


Single Backplane Configuration

Rules for Configuring Multiple Backplane Systems

- 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 20 ac loads (total). Unused ac loads from one backplane may not be added to another backplane if the second backplane loading will exceed 20 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 backplane must have an impedance of 120 ohms (obtained via the processor 220 ohm terminations and a separate 220 ohm terminator), and the last backplane must have a termination of 120 ohms.

- The cables(s) connecting the first two backplanes is 61 cm (2 ft) or greater in length.
- The cable(s) connecting the second backplane to the third backplane is 22 cm (4 ft) longer or shorter that the cable(s) connecting the first and second backplanes.
- The combined length of both cables cannot exceed 4.88 m (16 ft).
- The cables used must have a characteristic impedance of 120 ohms.


BACKPLANE WIRE 25.4 cm (10 in) MAX
220Ω
+
3.4V
−
ONE UNIT LOAD
ONE UNIT LOAD
220Ω
+
3.4V
−
20 AC LOADS MAX
PROCESSOR
CABLE/TERM
BACKPLANE WIRE 25.4 cm (10 in) MAX
ONE UNIT LOAD
ONE UNIT LOAD
CABLE
CABLE
20 AC LOADS MAX
ADDITIONAL CABLES & BACKPLANE
BACKPLANE WIRE 25.4 cm (10 in) MAX
ONE UNIT LOAD
ONE UNIT LOAD
120Ω
+
3.4V
−
CABLE
20 AC LOADS MAX
TERM
NOTES :
1. TWO CABLES (MAX.) 4.88m (16ft) (MAX.) TOTAL LENGTH.
2. 20 DC LOADS TOTAL (MAX)

Multiple Backplane Configuration

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 +5 V and +12 V power. Power requirements for each module are specified in the *Microcomputer Interface* 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.

The chart that follows illustrates the bus pin, it's mnemonic and description. The chart is defined by processor so that differences when they occur can be seen easily.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
AA 1	BIRQ5 L	BIRQ5 L	BIRQ5 L	Interrupt request priority level 5 (not implemented)	Interrupt request priority level 5
AB 1	BIRQ6 L	BIRQ6 L	BIRQ6 L	Interrupt request priority level 6 (not implemented)	Interrupt request priority 6
AC 1	BDAL16 L	BDAL16 L	BDAL16 L	Extended address bit (not implemented)	Address line 16 during addressing protocol; memory error data line during data transfer protocol.
AD 1	BDAL17 L	BDAL17 L	BDAL17 L	Extended address bit (not implemented)	Address line 17 during addressing protocol; memory error logic enable during data transfer protocol.
AE 1	SSI	SPARE	SSPARE1 (Alternate +5B)	Extended address bit (not implemented)	Special spare—not assigned or bused in DIGITAL cable or backplane assemblies; available for user connection. Optionally, this pin may be used for +5 V 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.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
AF 1	SRUN L	SSPARE2	SSPARE2	Special spare (not assigned, not bused, available for user interconnections)	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.
AH 1	SRUN L	SRUN L	SSPARE 3 SRUN	Special spare (not assigned, not bused, available for user interconnection)	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.
AJ 1	GND	GND	GND	Ground — System signal ground and dc return	Ground — System signal ground and dc return
AK 1	MSPAREA	MSPAREA	MSPAREA	Maintenance Spare — Normally connected together on the backplane at each option location (not bused connection)	Maintenance Spare — Normally connected together on the backplane at each option location (not bused connection)

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
AL 1	MSPAREB	MSPAREB	MSPAREA	Maintenance Spare — Normally connected together on the backplane at each option location (not bused connection)	
AM 1	GND	GND	GND	Ground — System signal ground and dc return	Ground — System signal ground and dc return
AN 1	BDMR L	BDMR L	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.	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.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
AP 1	BHALT L	BHALT L	BHALT L	Processor Halt — When BHALT L is asserted, the processor responds by halting normal program execution. External interrupts are ignored but memory refresh interrupts (enabled if W4 on M7264 and M7264-YA processor modules is removed) and DMA request/grant sequences are enabled. When in the halt state, the processor executes the ODT microcode and the console device operation is invoked.	Processor Halt — When BHALT L is asserted, the processor responds by going into console ODT mode.
AR 1	BREF L	BREF L	BREF L	Memory Refresh — Asserted by a processor microcode-generated refresh interrupt sequence (when enabled) or 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.	Memory Refresh—used to refresh dynamic memory devices. The LSI-11 processor microcode features automatic refresh control. BREF L is asserted during this time to override memory bank selection decoding. Interrupt requests are blocked out during this time.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
				CAUTION The user must avoid multiple DMA data transfers (burst or "hog" mode) that could delay refresh operation. Complete refresh cycles must occur once every 1.6 msec if required.	
AS 1	+12 B	+12 B	+ 5 B or +12 B	+12 V Battery Power — Secondary +12 V power connection. Battery power can be used with certain devices.	+12 Vdc or +5 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.
AT 1	GND	GND	GND	Ground — System signal ground and dc return	Ground — System signal ground and dc return
AU 1	PSPARE 1	PSPARE 1	PSPARE 1	Spare (Not assigned. Customer usage not recommended.)	Spare (Not assigned. Customer usage not recommended.)

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
AV 1	+5 B	+5 B	+5 B	+5 V Battery Power — Secondary +5 V power connection. Battery power can be used with certain devices.	+5 V Battery Power — Secondary +5 V power connection. Battery power can be used with certain devices. Not used on LSI-11/23
BA 1	BDCOK H	BDCOK H	BDCOK H	DC Power OK — Power supply-generated signal that is asserted when there is sufficient dc voltage available to sustain reliable system operation.	DC Power OK — Power supply-generated signal that is asserted when there is sufficient dc voltage available to sustain reliable system operation. This signal is also driven by the "wake-up" circuit on the LSI-11/23.
BB1	BPOK H	BPOK H	BPOK H	Power OK — Asserted by the power supply when primary power is normal. When negated during processor operation, a power fail trap sequence is initiated.	Power OK — Asserted by the power supply when primary power is normal. When negated during processor operation, a power fail trap sequence is initiated.
BC1	SSPARE4	SSPARE4	SSPARE4	Special spare (not assigned, not bused). Available for user interconnection except on M7264 or M7264-YA processor module location.	Special spare—not assigned or bused in DIGITAL cable and backplane assemblies; available for user interconnections.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
BD1	SSPARE5	SSPARE5	SSPARE5		Special spare—not assigned or bused in DIGITAL cable and backplane assemblies; available for user interconnections.
BE1	SSPARE6	SSPARE6	SSPARE6		Special spare—not assigned or bused in DIGITAL cable and backplane assemblies; available for user interconnections.
BF1	SSPARE7	SSPARE7	SSPARE7		Special spare—not assigned or bused in DIGITAL cable and backplane assemblies; available for user interconnections.
BH1	SSPARE8	SSPARE8	SSPARE8		Special spare—not assigned or bused in DIGITAL cable and backplane assemblies; available for user interconnections.
BJ1	GND	GND	GND	Ground — System signal ground and dc return.	Ground — System signal ground and dc return.
BK1 BL1	MSPAREB MSPAREB	MSPAREB MSPAREB	MSPAREB MSPAREB	MAINTENANCE Spare — Normally connected together on the backplane at each option location (not a bused connection).	MAINTENANCE Spare — Normally connected together on the backplane at each option location (not a bused connection).

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
BM1	GND	GND	GND	Ground — System signal ground and dc return.	Ground — System signal ground and dc return.
BN1	BSACK L	BSACK L	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.	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	BIRQ7 L	BIRQ7 L	Interrupt request priority level 7 (not implemented)	Interrupt request priority level 7
BR1	BEVNT L	BEVNT L	BEVNT L	External Event Interrupt Request — When asserted, the processor responds (if PS bit 7 is 0) by entering a service routine via vector address 100_8. A typical use of this signal is a line time clock interrupt.	External Event Interrupt Request — When asserted, the processor responds (if PS bit 7 is 0) by entering a service routine via vector address 100_8. A typical use of this signal is a line time clock interrupt.
BS1	PSPARE 4	PSPARE 4	+12 B	Spare (Not assigned. Customer usage not recommended).	+12 Vdc battery backup power (not bused to AS1 in all DIGITAL backplanes)
BT1	GND	GND	GND	Ground — System signal ground and dc return.	Ground — System signal ground and dc return.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
BU1	PSPARE2	PSPARE2	PSPARE2	Spare (Not assigned. Customer usage not recommended.)	Power spare 2 (not assigned a function, not recommended for use). If a module is using −12 V (on pin AB2) and if the module is accidentally inserted backwards in the backplane, −12 Vdc appears on pin BU1.
BV1	+5	+5	+5	+5 V Power—Normal +5 V dc system power.	+5 V Power—Normal +5 V dc system power.
AA2	+5	+5	+5	+5 V Power—Normal +5 V dc system power.	+5 V Power—Normal +5 V dc system power.
AB2	−12	−12	−12	−12 V Power — −12 V dc (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 required voltages(s) hence, −12 V power is not required with DIGITAL-supplied options.	−12 V Power — −12 V dc (optional) power for devices requiring this voltage. **NOTE** Modules that require negative voltages contain an inverter circuit (on each module) which generates the required voltages(s). Hence, −12 V power is not required with DIGITAL-supplied options including the KDF11-AA processor.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
AC2	GND	GND	GND	Ground — System signal ground and dc return.	Ground — System signal ground and dc ret‿rn.
AD2	+12	+12	+12	+12 V Power — 12 V dc system power.	+12 V Power — 12 V dc system power.
AE2	BDOUT L	BDOUT L	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 BRPL L to complete the transfer.	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 BRPL L to complete the transfer.
AF2	BRPLY L	BRPLY L	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.	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.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
AH2	BDIN L	BDIN L	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 L, it indicates that an interrrupt operation is occurring. The master device must deskew input data from BRPLY 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 L, it indicates that an interrrupt operation is occurring. The master device must deskew input data from BRPLY L.
AJ2	BSYNC L	BSYNC L	BSYNC L	Synchronize — BSYNC L is asserted by the bus master device to indicate that it has placed an address on BDAL	Synchronize — BSYNC L is asserted by the bus master device to indicate that it has placed an address on BDAL <0:17> L.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
				<0:17> L. The transfer is in process until BSYNC L is negated.	The transfer is in process until BSYNC L is negated.
AK2	BWTBT L	BWTBT L	BWTBT L	Write/Byte — BWTBT L is used in two ways to control a bus cycle: It is asserted during 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.	Write/Byte — BWTBT L is used in two ways to control a bus cycle: It is asserted during 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	BIRQ4 L	BIRQ4 L	Interrupt Request — A 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.	Interrupt request priority level 4

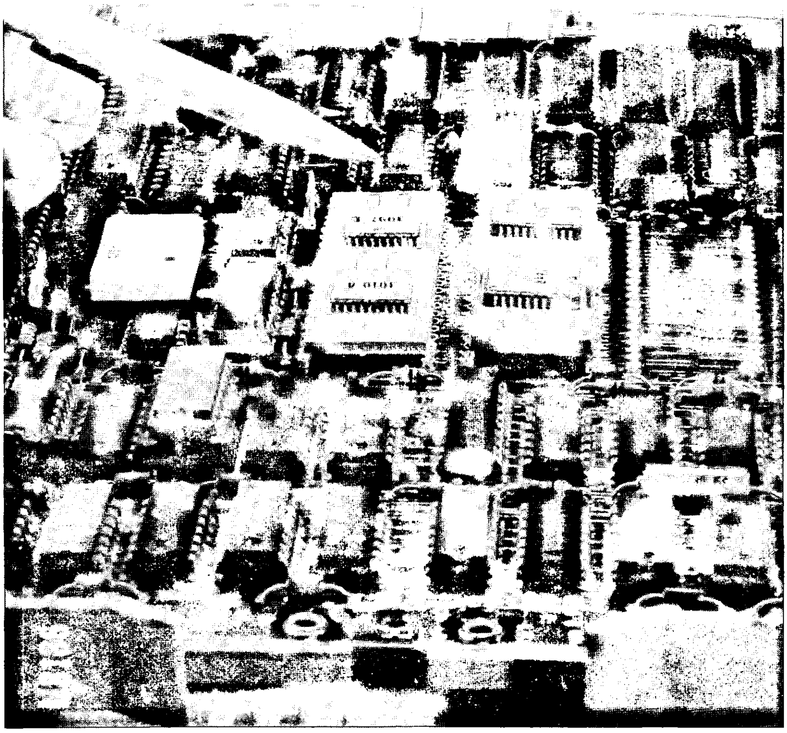
BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
AM2 AN2	BIAKI L BIAKO L	BIAKI L BIAKO L	BIAKO L BIAKO L	Interrupt Acknowledge and Interrupt Acknowledge Output — This is an interrupt acknowledge signal which is generated by the processor in response to an interrupt request (BIRQ L). The processor asserts BIAKO L, which is routed to the BIAKI L pin of the first device on the bus. If it is requesting an interrupt, it will inhibit passing BIAKO L. If it is not asserting BIRQ L, the device will pass BIAKI L to the next (lower priority) device via its BIAKO L pin and the lower priority device BIAKI L pin.	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 next priority device (electrically closest to the processor). This device accepts the Interrupt Acknowledge under two conditions: 1) The device requested the bus by asserting an interrupt, 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.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
AP2	BBS7 L	BBS7 L	BBS7 L	Bank 7 Select — The bus master asserts BBS7 L when an I/O device address in the upper 4K address range is placed on the bus. BSYNC L is then asserted and BBS7 L remains active for the duration of the addressing portion of the bus cycle.	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/O page.
AR2 AS2	BDMGI L BDMGO L	BDMGI L BDMGO L	BDMGI L BDMGO L	DMA Grant Input and DMA Grant Output — This is the processor-generated daisy-chained signal which grants bus mastership to the highest priority DMA device along the bus. The processor generates BDMGO L, which is routed to the BDMGI L pin of the first device on the bus. If it is requesting the bus, it will inhibit passing BDMGO L. If it is not requesting the bus, it will pass the BDMGI L signal to the next (lower pri-	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 not, the device passes the grant (as-

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
				ority) device via its BDMGO L pin. The device asserting BDMR L is the device requesting the bus, and it responds to the BDMGI L signal by negating BDMR, asserting BSACK L, assuming bus mastership, and executing the required bus cycle. **CAUTION** DMA device transfers must not interfere with the memory refresh cycle.	serts BDMGO L) to the next device on the bus. This process continues until the requesting device acknowledges the grant.
AT2	BINIT L	BINIT L	BINIT L	Initialize — BINIT is asserted by the processor to initialize or clear all devices connected to the I/O bus. The signal is generated in response to a power-up condition (the negated condition of BDCOK H), or by executing a RESET instruction.	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 programming and engineering specifications for the device.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
AU2 AV2	BDAL0 L BDAL1 L	BDAL0 L BDAL1 L	BDAL0 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.	Data/Address Lines — These two lines are part of the 8-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	+5	+5	+5 V Power—Normal +5 V dc system power.	+5 V Power—Normal +5 V dc system power.
BB2	−12	−12	−12	−12 V Power — −12 V dc (optional) power for devices requirimg this voltage.	−12 V Power — −12 V dc (optional) power for devices requiring this voltage. Not used by LSI-11/23.
BC2	GND	GND	GND	Ground — System signal ground and dc return.	Ground — System signal ground and dc return.
BD2	+12	+12	+12	+12 V Power — +12 V system power.	+12 V Power — +12 V system power.

BUS PIN	DEC BUS MNEMONICS	LSI-11/2 MNEMONICS	LSI-11/23 MNEMONICS	DESCRIPTION LSI-11/2	DESCRIPTION LSI-11/23
BE2	BDAL2 L	BDAL2 L	BDAL2 L	Data/Address Lines — These 14 lines are part of the 16-line data/address bus previously described.	Data/Address Lines — These 14 lines are part of the 16-line data/address bus previously described.
BF2	BDAL3 L	BDAL3 L	BDAL3 L		
BH2	BDAL4 L	BDAL4 L	BDAL4 L		
BJ2	BDAL5 L	BDAL5 L	BDAL5 L		
BK2	BDAL6 L	BDAL6 L	BDAL6 L		
BL2	BDAL7 L	BDAL7 L	BDAL7 L		
BM2	BDAL8 L	BDAL8 L	BDAL8 L		
BN2	BDAL9 L	BDAL9 L	BDAL9 L		
BP2	BDAL10 L	BDAL10 L	BDAL10 L		
BR2	BDAL11 L	BDAL11 L	BDAL11 L		
BS2	BDAL12 L	BDAL12 L	BDAL12 L		
BT2	BDAL13 L	BDAL13 L	BDAL13 L		
BU2	BDAL14 L	BDAL14 L	BDAL14 L		
BV2	BDAL15 L	BDAL15 L	BDAL15 L		



CHAPTER 9
MEMORY MANAGEMENT

The PDP-11/23 processor implements a 256K byte physical address space. This improves the 64K byte maximum physical address space previously available in LSI-11 processors. The mapping, or translation, of 16-bit virtual addresses to 18-bit physical addresses is implemented in one MOS/LSI integrated circuit. This chip is called the memory management unit, MMU. The memory management functionality is software-compatible with other PDP-11 processors, the PDP-11/34, -11/60 and -11/70. Sixteen programmable relocation registers (eight for kernel mode and eight for user mode) are used to accomplish the mapping function. The contents of these registers are combined with the 16-bit virtual address to form an 18-bit physical address. The actual transformation occurs transparently to an executing program.

The memory management chip is designed for use in a single or multiprogramming environment. The processor can operate in two modes: kernel and user.

In **kernel mode,** the software has complete control and can execute all instructions. Monitors and supervisory programs are executed in kernel mode.

In a multiprogramming environment, several user programs are resident in memory at any given time. Then the kernel software normally accomplishes the following:

- control of execution of the various user programs
- allocation of memory and peripheral device resources
- safeguard of the integrity of the system as a whole by careful control of each user program

In **user mode,** the software is executed in a restricted environment and is prevented from executing certain instructions that could be destructive to the software system. For example: modification of the kernel program, halting the computer, or using memory space assigned to the kernel or other users.

In a multiprogramming system, the kernel software using the memory management unit assigns pages (relocatable memory segments) to a user's program and prevents the user from making any unauthorized access to those pages outside the assigned area. A user can thus effectively be prevented from accidental or willful destruction of any other user program or of the system executive program.

Hardware-implemented features enable the operating system to dynamically allocate memory upon demand while a program is being run.

Basic Addressing

The PDP-11 family word length is 16 bits; however, the LSI-11 bus and the 11/23 addressing logic are 18 bits wide. While a 16-bit word can generate virtual address references up to 32K words (64K bytes), the LSI-11/23 and the LSI-11 bus can reference 18-bit physical addresses up to 128K words, (256K bytes). The extra two bits of addressing logic provide the basic framework for expanding memory references.

8K bytes of address space are reserved for the I/O device registers. Thus 248K bytes remain for the main memory.

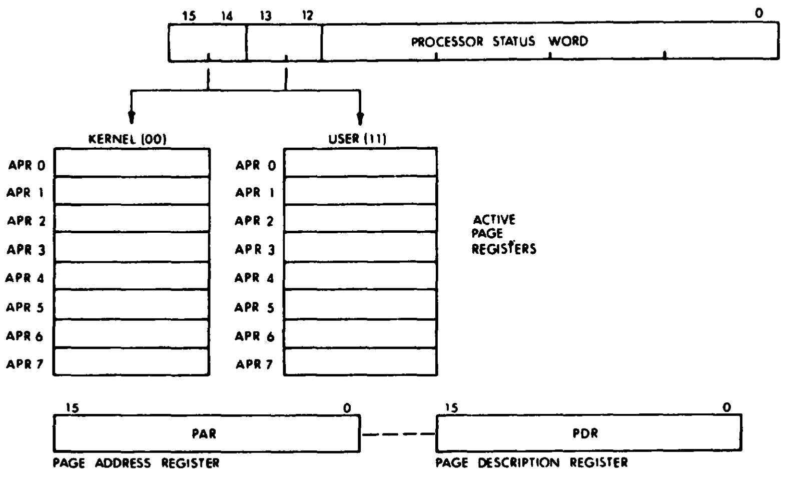
Active Page Registers

The memory management unit uses two sets of eight 32-bit active page registers (APR). An APR is actually a pair of 16-bit registers: a page address register (PAR) and a page descriptor register (PDR). These registers are always used as a pair and contain all the information needed to describe and relocate the currently active memory pages.

One set of APRs is used in **kernel** mode, and the other in **user** mode. The set to be used is determined by the current CPU mode contained in the processor status word, bits 15 and 14.

Capabilities Provided by Memory Management

Memory Size:	256K bytes (248K bytes plus 8K bytes for the I/O Page)
Address Space:	Virtual 64K bytes (16 bits) Physical 256K bytes (18 bits)
Modes of Operation:	Kernel and User
Stack Pointers:	Two, one for each mode
Number of Pages:	16 (eight for each mode)
Page Length:	32 to 4,096 words, 64 to 8, 192 bytes
Memory Page Protection:	No access Read-only Read/write



Active Page Registers

MEMORY RELOCATION

When the memory management unit is operating, the normal 16-bit direct byte address is no longer interpreted as a direct physical address (PA) but as a virtual address (VA) containing information to be used in constructing a new 18-bit physical address. The information contained in the virtual address is combined with relocation and description information contained in the active page register to yield an 18-bit physical address.

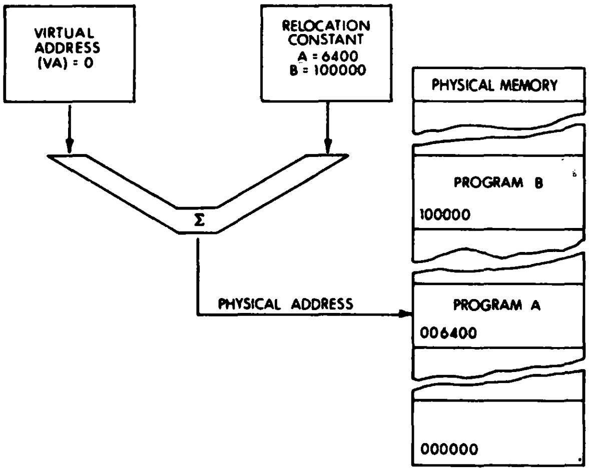
Because addresses are relocated automatically, the computer may be considered to be operating in virtual address space. This means that regardless of where a program is loaded into physical memory, it will not have to be relinked; it always appears to be at the same virtual location in memory.

The virtual address space is divided into eight 8K-byte pages. Each page is relocated separately. This useful feature in multiprogrammed timesharing systems permits a new large program to be loaded into discontinuous blocks of physical memory.

A basic function of the MMU is to perform memory relocation and to provide extended memory addressing capability for systems with more than 64K bytes of physical memory. Two sets of page address registers are used to relocate virtual addresses to physical addresses in memory. These sets are used as hardware relocation registers that permit several users' programs, each starting at virtual address 0, to reside simultaneously in physical memory.

Program Relocation

The page address registers are used to determine the starting physical address of each relocated program in physical memory. The following figure shows a simplified example of the relocation concept.



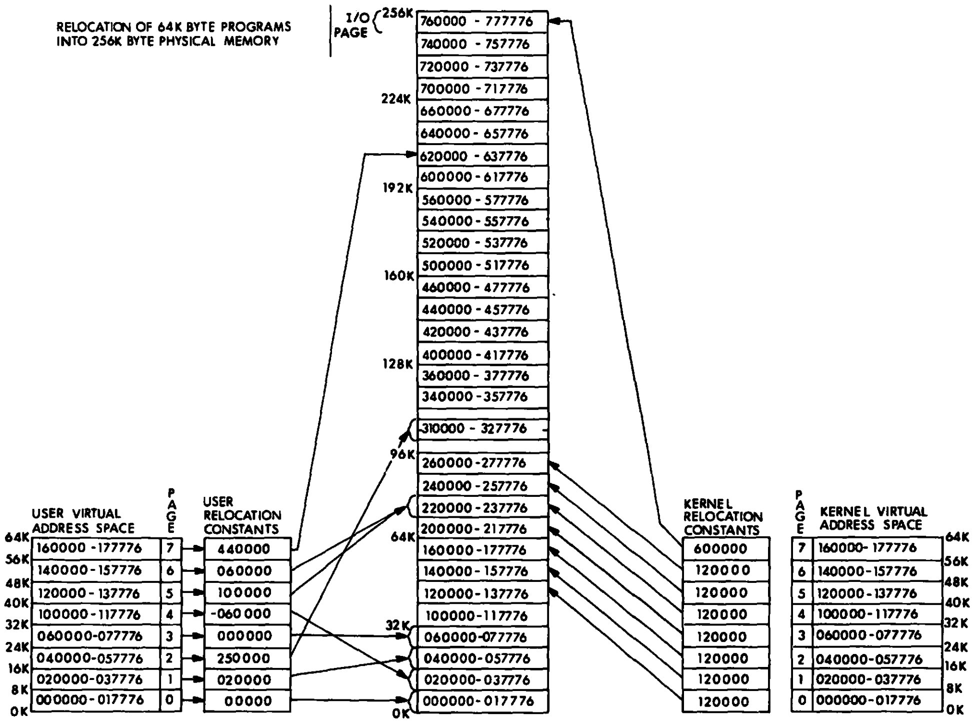
Simplified Memory Relocation

Program A, virtual starting address 0, is relocated by a constant to provide physical address 6400_8.

If the program's next virtual address is 2, the relocation constant will then cause physical address 6402_8, which is the second item of program A, to be accessed. When program B is running, the relocation constant is changed to 100000_8. The program B virtual addresses starting at 0 are relocated to access physical addresses starting at 100000_8. Using the active page address registers to provide relocation eliminates the need to relink a program each time it is loaded into a different physical memory location. The program always appears to start at the same address.

A program is relocated in pages consisiting of from 1 to 128 blocks. Each block is 64 bytes in length. Thus, the maximum length of a page is 8,192 (128 × 64) bytes. Using all of the eight available active page registers in a set, a maximum program length of 65,536 bytes can be accommodated. Each of the eight pages can be relocated anywhere in the physical memory, as long as each relocated page begins on a boundary that is a multiple of 64 bytes. However, for pages that are smaller than 8K bytes, only the memory actually allocated to the page may be accessed.

The relocation example shown in the accompanying figure illustrates several points about memory relocation.



Relocation of a 64K-byte Program into 256K-byte Physical Memory

Although the user program appears to the processor to be in contiguous address space, the 64K byte virtual address space has been scattered throughout the physical memory space. It has been segmented, split into 8K byte pages. And it has been relocated, each of the pages assigned various positions in physical memory.

Pages may be relocated to higher, lower, or the same physical address with respect to their virtual address range. User pages 1, 2, 5, 6, and 7 have been relocated to a higher range of physical addresses. User page 4 has been relocated to a lower range, and user pages 0 and 3 have not been relocated.

For simplicity, all pages except user page 2, have been relocated on an 8K byte boundary. User page 2 has been relocated on a 4K byte boundry. Pages may be relocated to any 64 byte boundary.

Each page is relocated independently. There is no reason two or more pages could not be relocated to the same physical memory space. Using more than one page address register in the set to access the same space would be one way of providing different memory access rights to the same data, depending on which part of the program was referencing that data. User pages 5 and 6 are relocated to the same physical memory. Kernel page 4 is also relocated to the same physical memory as user pages 5 and 6. This might provide two programs access to a common data or communications space.

The kernel virtual address space has simply been shifted up 20K words and still remains contiguous in physical memory.

The uppermost 8K-byte page of physical memory is designated as the I/O page. The uppermost 8K bytes of the kernel address space has been relocated to the I/O page. Therefore, only the kernel and not the user program has access to the peripheral devices located in the I/O page.

The RELOCATION CONSTANT used in the preceding discussion is derived from the contents of the appropriate PAR register.

Memory Units

Block:	64 bytes
Page:	1 to 128 blocks (64 to 8,192 bytes)
No. of pages:	8 per mode
Size of relocatable memory	65,536 bytes, max (8 × 8,192)

PROTECTION

A timesharing system must perform multiprogramming; i.e., it allows several programs to reside in memory simultaneously and to execute sequentially. Access to these programs, and the memory space they occupy, must be strictly defined and controlled. A timesharing system, therefore, requires several types of memory protection.

- User programs must not be allowed to expand beyond their allocated space unless authorized by the system manager.
- Users must be prevented from modifying common subroutines and algorithms that are resident for all users.
- Users must be prevented from gaining control of, or modifying, the operating system software
- Users must be prevented from accessing or modifying memory occupied by other users.

Memory management provides the hardware facilities to implement all the above types of memory protection.

Inaccessible Memory

Each page has a 2-bit access control key associated with it. The key is part of the page descriptor register (PDR). The key is assigned under operating system control. When the key is set to 0, the page is defined as nonresident. Any attempt by a user program to access a nonresident page is prevented by an immediate abort of the offending instruction. Abort means that execution of the instruction ceases immediately instead of waiting for the end of the instruction to report the error. Using this feature to provide memory protection, only those pages associated with the current program are set to legal access keys. The access control keys of all other program pages are set to 0, which prevents illegal memory references.

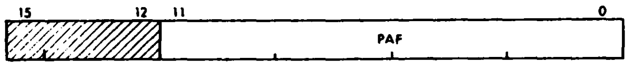
Read-Only Memory

The access control key for a page can be set to 2, which allows read (fetch) memory references to the page, but immediately halts any attempt to write into that page. This read-only type of memory protection can be afforded to pages that contain common data, subroutines, or shared algorithms. This type of memory protection allows the access rights to a given memory area to be user-dependent. That is, the access right to a memory area may be varied for different users by altering the access control key.

A page address register in each of the sets (kernel and user modes) may be set up to reference the same physical page in memory and each may be keyed for different access rights. For example, the user access control key might be 2 (read-only access for user programs), and the kernel access control key might be 4 (allowing complete read/write access for the operating system).

Multiple Address Space
There are two complete PAR/PDR sets provided: one set of registers for kernel mode and one set for user mode. This affords the operating system software another type of memory protection. The mode of operation is specified by the processor status word current mode field, or previous mode field, as determined by the current instruction. Each mode has its own corresponding stack pointer (R6) for protection, as well as software considerations.

A user mode program is relocated by its own PAR/PDR set, as is a kernel program. This makes it impossible for a program running in one mode to accidentally reference space allocated to another mode when the active page registers are set correctly. For example, a user cannot transfer to kernel space. The kernel mode address space may be reserved for resident system monitor functions, such as the basic input/output control routines, memory management trap handlers, and timesharing scheduling modules. By dividing the types of timesharing systems functionally between the kernel and user modes, a minimum of space control housekeeping is required as the time-shared operating system sequences from one user program to the next. For example, only the user PAR/PDR set needs to be updated as each new user program is serviced.



PSW

Mode Specification in Processor Status Word — PS<15:14> specify the current memory management mode. These bits are used to select the corresponding PAR/PDR set to be used for the currently executing program. PS<13:12> specify the previous memory management mode. These bits are used by the memory management instructions to commmunicate between kernel and user address spaces. When an implicit mode change occurs, the previous mode bits (PS<13:12>) are loaded by hardware with the contents of the current mode bits (PS<15:14>). This change can occur whenever an interrupt or trap is processed. PS<15:14> are cleared when power is applied. Clearing these bits selects kernel mode. PS<15:12> are encoded as shown below.

PS15:14 or PS13:12	PAR/PDR Set Enable	Stack Pointer Selected
00	Kernel	Kernel (KSP)
01	Reserved for future DIGITAL use. Specifies supervisor mode on some PDP-11s. Does not cause an abort.	Supervisor (SSP) - Reserved for future DIGITAL use.
10	Illegal. Does not cause an abort.	Reserved for future DIGITAL use.
11	User	User (USP)

Each mode selects its own corresponding stack pointer. Thus, all program references to register R6 use the stack pointer register as specified by PS<15:14>. Stack pointer selection occurs whether the memory management unit is enabled or not (SR0 bit 0 is a 1). The different stack pointers are initialized by first loading the appropriate mode value in PS<15:14>, and can be examined by console ODT.

Processor Status Word Protection — There are various software methods of affecting PS<15:00>. Since kernel mode is defined to allow software access to all hardware features, free access to the PS is allowed. Since user mode is defined for operating user programs and thus protecting the operating system software, certain PS bits, such as the mode and priority level fields, are protected.

Processor Status Word Protection

	RTI,RTT		Traps & Interrupts		Explicit PS Access		MTPS		Power Up
PS Bits	User	Kernel	User	Kernel	User	Kernel	User	Kernel	
Condition Code PS<3:0>	Loaded From Stack	Loaded From Stack	Loaded From Vector	Loaded From Vector	Loaded From Source	Loaded From Source	Loaded From Source	Loaded From Source	Cleared
Trap Bit PS4	Loaded From Stack	Loaded From Stack	Loaded From Vector	Loaded From Vector	Unchanged	Unchanged	Unchanged	Unchanged	Cleared
Interrupt Priority PS<7:5>	Unchanged	Loaded From Stack	Loaded From Vector	Loaded From Vector	Loaded From Source	Loaded From Source	Unchanged	Loaded From Source	SET when powered up to bootstrap Cleared when powered up to ODT, loc 24, or microcode
SI PS 8	Loaded From Stack	Loaded From Stack	Loaded From Vector	Loaded From Vector	Loaded From Source	Loaded From Source	Non-Accessible	Non-Accessible	Cleared
Previous Mode PS<13:12>	Unchanged	Loaded From Stack	Copied From PS <15:14>	Copied From PS <15:14>	Loaded From Source	Loaded From Source	Non-accessible	Non-Accessible	Cleared
Current Mode PS<15:14>	Unchanged	Loaded From Stack	Loaded From Vector	Loaded From Vector	Loaded From Source	Loaded From Source	Non-accessible	Non-accessible	Cleared

User Mode Restrictions — User mode is intended for executing user programs. While in user mode, the program is restricted from using those hardware features that could disrupt system integrity. The following hardware features are protected in user mode.

HALT instruction—Instead of entering console ODT, a HALT instruction causes a trap to kernel location 10_8. The intent is not to allow a user program to halt the operating system.

RESET instruction—Instead of causing a BUS initialize, a RESET instruction is executed as an NOP instruction. The intent is to prevent the user program from initializing I/O devices.

The MTPS instruction, when executed in user mode allows access to PS<03:00> only. All other PS bits are vital to system operations and cannot be affected. Explicit access to the PS in user mode allows full access to the PS (except T bit), however the kernel program has control via the mapping registers over the mapping of a user program into the PS address.

Interrupt and Trap Processing — All interrupt and trap vectors are always used in kernel mode when the new PC and PS are fetched. The processor's first step in processing the interrupt or trap is to fetch the new PS value from the interrupt or trap location plus two. This determines which mode, and consequently which stack pointer, to use for pushing the old PC and PS. The 11/23 copies the old PS into a temporary register and then loads the new PS value. PS<15:14> are loaded from the memory location to select the new current mode. PS<13:12> (previous mode) are loaded with the old value in PS<15:14>, to keep a record of what the previous mode was. This is the only place where the PS previous mode bits copy the current mode bits.

This process allows communication between mode address spaces using the memory management instructions. The remaining PS bits are loaded from the memory location. Thus, interrupt and trap service routines can be executed in either kernel or user mode, depending on the contents of the vector plus two locations.

PAR/PDR Address Assignments

Kernel Active Page Registers			User Active Page Registers		
No.	PAR	PDR	No.	PAR	PDR
0	772340	772300	0	777640	777600
1	772342	772302	1	777642	777602
2	772344	772304	2	777644	777604
3	772346	772306	3	777646	777606

Kernel Active Page Registers			User Active Page Registers		
No.	PAR	PDR	No.	PAR	PDR
4	772350	772310	4	777650	777610
5	772352	772312	5	777652	777612
6	772354	772314	6	777654	777614
7	772356	772316	7	777656	777616

PAGE ADDRESS REGISTER (PAR)

The page address register (PAR) contains the 12-bit page address field (PAF) that specifies the base address of the page.

Bits 15-12 are implemented but are reserved for future DIGITAL use.

The page address register can be considered a relocation constant, or a base register containing a base address. Either interpretation indicates the basic function of the page address register (PAR) in the relocation scheme.

PAGE DESCRIPTOR REGISTER (PDR)

The page descriptor register (PDR) contains information pertinent to page expansion, page length, and access control.

Access Control Field (ACF)

This 2-bit field, bits 2 and 1 of the PDR, describes the access rights to this particular page. The access bits specify the manner in which a page may be accessed and whether or not a given access should result in an abort of the current operation. A memory reference that causes an abort is not completed and is terminated immediately. Aborts are caused by attempts to access nonresident pages, page length errors, or by access violations such as attempts to write into a read-only page.

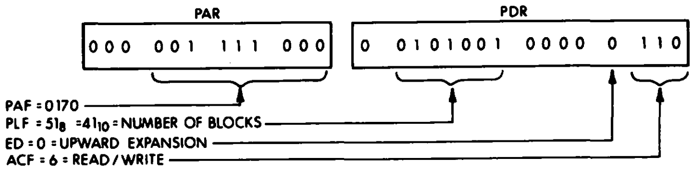
In the context of access control, the term "write" indicates the action of any instruction that modifies the contents of any addressable word. The accompanying table lists the ACF keys and their functions. The ACF is written into the PDR under program control.

Access Control Field Keys

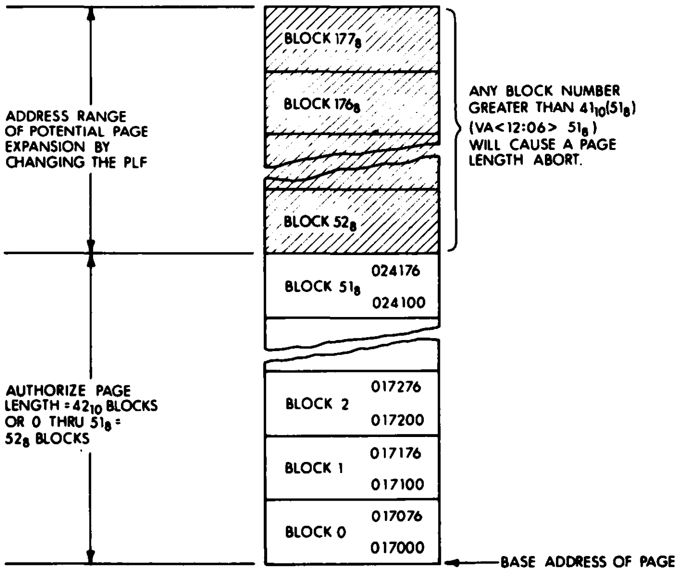
ACF	Key	Description	Function
00	0	Nonresident	Abort any attempt to access this nonresident page

ACF	Key	Description	Function
01	2	Resident read-only	Abort any attempt to write into this page.
10	4	Unused	Abort all accesses
11	6	Resident read/write	Read or write allowed. No abort occurs.

Expansion Direction (ED)
The ED bit located in PDR bit position 3 indicates the authorized direction in which the page can expand. A logic 0 in the bit (ED = 0) indicates the page can expand upward from relative zero. A logic 1 in this bit (ED = 1) indicates the page can expand downward toward relative zero. The ED bit is written into the PDR under program control. When the expansion direction is upward (ED = 0), the page length is increased by adding blocks with higher relative addresses. Upward expansion is usually specified for program or data pages to add more program or table space. An example of page expansion **upward** is shown in the accompanying figure.

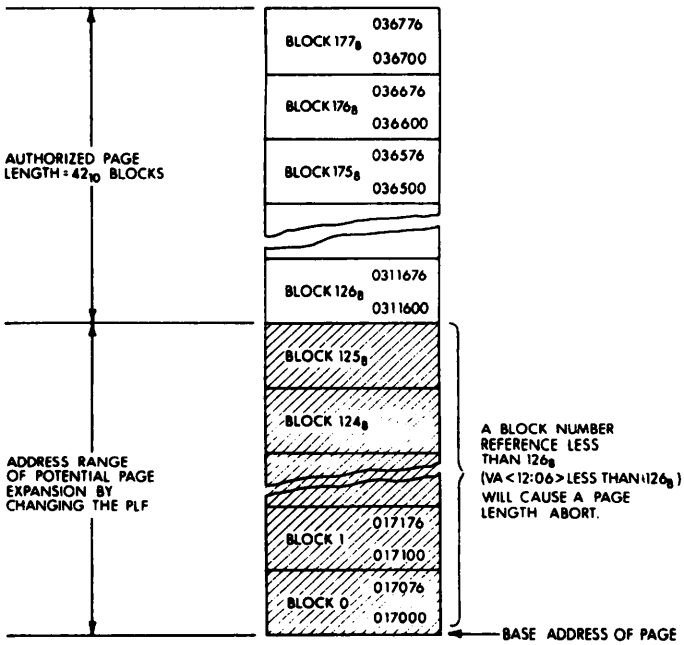


NOTE: To specify a block length of 42 for an upward expandable page, write highest authorized block number directly into high byte of PDR. Bit 15 is not used because the highest allowable block number is 177_8.


BLOCK 177₈
BLOCK 176₈
BLOCK 52₈
ADDRESS RANGE OF POTENTIAL PAGE EXPANSION BY CHANGING THE PLF
ANY BLOCK NUMBER GREATER THAN 41₁₀(51₈) (VA<12:06> 51₈) WILL CAUSE A PAGE LENGTH ABORT.
BLOCK 51₈
024176
024100
AUTHORIZE PAGE LENGTH = 42₁₀ BLOCKS OR 0 THRU 51₈ = 52₈ BLOCKS
BLOCK 2
017276
017200
BLOCK 1
017176
017100
BLOCK 0
017076
017000
BASE ADDRESS OF PAGE

Example of an Upward-Expandable Page

When the expansion direction is downward (ED = 1), the page length is increased by adding blocks with lower relative addresses. Downward expansion is specified for stack pages so that more stack space can be added. An example of page expansion **downward** is shown in the following figure.


AUTHORIZED PAGE LENGTH = 42₁₀ BLOCKS
ADDRESS RANGE OF POTENTIAL PAGE EXPANSION BY CHANGING THE PLF
BLOCK 177₈
036776
036700
BLOCK 176₈
036676
036600
BLOCK 175₈
036576
036500
BLOCK 126₈
0311676
0311600
BLOCK 125₈
BLOCK 124₈
BLOCK 1
017176
017100
BLOCK 0
017076
017000
A BLOCK NUMBER REFERENCE LESS THAN 126₈ (VA<12:06> LESS THAN 126₈) WILL CAUSE A PAGE LENGTH ABORT.
BASE ADDRESS OF PAGE

Example of a Downward-Expandable Page

Written Into (W)
The W bit located in PDR bit position 6 indicates whether the page has been written into since it was loaded into memory. W = 1 is affirmative. The W bit is automatically cleared when the PAR or PDR of that page is written into. It can be set only by the control logic.

In disk swapping and memory overlay applications, the W bit (bit 6) can be used to determine which pages in memory have been modified by a user. Those that have been written into must be saved in their current form. Those that have not been written into (W = 0) need not be saved and can be overlayed with new pages, if necessary.

Page Length Field (PLF)
The 7-bit PLF located in PDR<14:08> specifies the authorized length of the page in 32-word blocks. The PLF holds block numbers from 0 to 177_8, thus allowing any page length from 1 to 128_{10} blocks. The PLF is written into the PDR under program control. The PLF contains the block number of the last accessible block in the virtual page.

PLF For an Upward-Expandable Page — When the page expands upward, the PLF must be set to one less than the intended number of blocks authorized for that page. This is the block number of the highest block authorized for the virtual page. For example, if 52_8 (42_{10}) blocks are authorized, the PLF is set to 51_8 (41_{10}). The hardware compares the virtual address block number, VA<12:06> with the PLF to determine if the virtual address is within the authorized page length.

When the virtual address block number is less than or equal to the PLF, the virtual address is within the authorized page length. If the virtual address is greater than the PLF, a page length fault (address too high) is detected by the hardware and a halt occurs. In this case, the virtual address space legal to the program is noncontiguous because the three most significant bits of the virtual address are used to select the PAR/PDR set.

PLF For a Downward-Expandable Page — The capability of providing downward expansion for a page is intended specifically for those pages that are to be used as stacks. A stack starts at the highest location reserved for it and expands downward toward the lowest address as items are added to the stack.

When the page is to be downward expandable, the PLF must contain the block number of the lowest authorized block for the virtual page. This number is the 2's complement of the number of blocks required.

In this example, a 42-block page is required.

PLF is derived as follows:

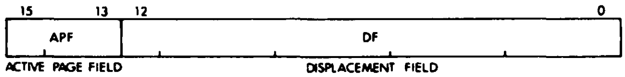
$42_{10} = 52_8$; 2's complement = 126_8

The same PAF is used in both examples shown here. This is done to emphasize that the PAF, as the base address, always determines the lowest address of the page, whether it is upward- or downward-expandable.

VIRTUAL AND PHYSICAL ADDRESSES

Construction of a Physical Address

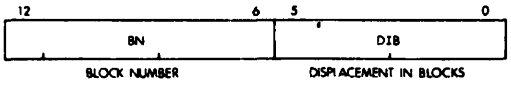
The basic information needed for the construction of a physical address (PA) comes from the virtual address (VA), and the appropriate APR set.



Interpretation of a Virtual Address

The virtual address consists of the following:

- the active page field (APF) — This 3-bit field determines which of the eight active page registers (APR0-APR7) will be used to form the physical address
- the displacement field (DF) — This 13-bit field contains an address relative to the beginning of a page. This permits page lengths up to 4K words (2^{13} = 8K bytes). The DF is further subdivided into two fields.



Displacement Field of Virtual Address

The displacement field (DF) consists of the following:

- the block number (BN). This 7-bit field is interpreted as the block number within the current page.
- the displacement in block (DIB). This 6-bit field contains the displacement within the block referred to by the block number.

The remainder of the information needed to construct the physical address comes from the 12-bit page address field (PAF), part of the active page register, and specifies the starting address of the memory which that APR describes. The PAF is actually a block number in the physical memory; PAF = 3 indicates a starting address of word 96 (3 × 32 = 96) in physical memory.

The logical sequence involved in constructing a physical address is as follows:

- Select a set of active page registers depending on current mode specified by PS<15:14>. The active page field of the virtual address is used to select an active page register (APR0-APR7).
- The page address field of the selected APR contains the starting address of the currently active page as a block number in physical memory.
- The block number from the virtual address is added to the block number from the page address field to yield the number of the block in physical memory which will contain the physical address being constructed.
- The displacement in blocks from the displacement field of the virtual address is joined to the physical block number to yield an 18-bit physical address.

Determining the Program Physical Address

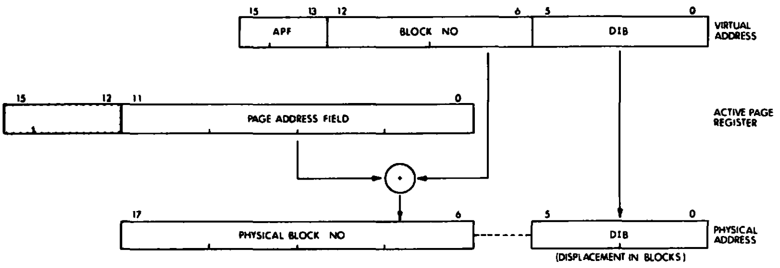
A 16-bit virtual address can specify up to 64K bytes, in the range from 000000_8 to 177776_8 (word boundaries are even numbers). The three most significant virtual address bits designate the PAR/PDR pair to be referenced during page address relocation. The following table lists the virtual address ranges that specify each of the PAR/PDR sets.

Relating Virtual Address to PAR/PDR Set

Virtual Address Range	PAR/PDR Set
000000-17776	0
020000-37776	1
040000-57776	2
060000-77776	3
100000-117776	4
120000-137776	5
140000-157776	6
160000-177776	7

NOTE

Any use of page lengths of less than 8K bytes causes unaddressable holes in the virtual address space.



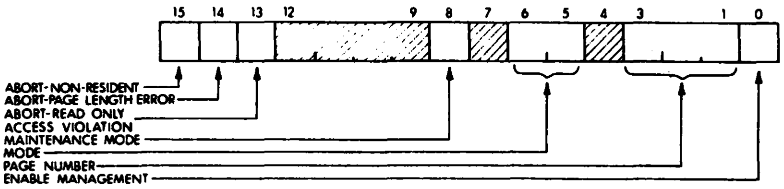
Construction of a Physical Address

STATUS REGISTERS

Aborts generated by protection hardware are vectored through kernel virtual address 250_8. Status registers are used to determine why the abort occurred. Note that an abort to a location which is itself an invalid address will cause another abort. Thus, the kernel program must ensure that kernel virtual address 250_8 is mapped into a valid physical address, otherwise an infinite loop requiring operator intervention will occur.

Status Register 0 (SR0)

SR0 contains abort error flags, memory management enable, and other essential information required by an operating system to recover from an abort. The SR0 format shown has as its address 777572_8> This register is cleared by application of power or a RESET instruction.



Format of Status Register 0 (SR0)

Bits 15-13 are the abort flags. They may be considered to be in priority order in that flags to the right are less significant and should be ignored. For example, a nonresident abort service routine would ignore page length and access control flags. A page length abort service routine would ignore an access control fault.

Bit 15, 14, or 13 when set (abort conditions) causes the logic to freeze status register SR2. This is done to facilitate recovery from the abort.

Note that only SR0 bit 0 can be set under program control to provide memory management control information. Only that information which is automatically written into the remaining bits as a result of hardware actions is useful as a monitor of the status of the memory management unit. Setting bits 15-13 under program control will not cause traps to occur. These bits, however, must be reset to 0 by software after an abort has occurred in order to resume monitoring memory management.

Abort Nonresident — Bit 15 is the abort nonresident bit. It is set by attempting to access a page with an access control field (ACF) key equal to 0 or 4.

Abort Page Length — Bit 14 is the abort page-length bit. It is set by attempting to access a location in a page with a block number (virtual address bits 12-6) that is outside the area authorized by the page length field (PLF) of the PDR for that page.

Abort Read-Only — Bit 13 is the abort read-only bit. It is set by attempting to write in a read-only page, access key 2.

There are no restrictions against abort bits being set simultaneously by the same access attempt.

Mode of Operation — Bits 5 and 6 indicate the CPU mode (user or kernel) associated with the page causing the abort (kernel = 00, user = 11).

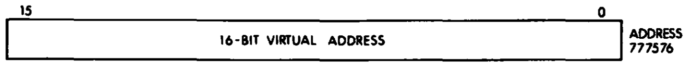
Page Number — Bits 3-1 contain the virtual page number referenced. Pages, like blocks, are numbered from 0 upwards. The page number field is used by the error recovery routine to identify the page being accessed if an abort occurs.

Enable Relocation and Protection — Bit 0 is the enable bit. When it is 1, all addresses are relocated and protected by the memory management unit. When bit 0 is set to 0, the memory management unit is disabled and addresses are neither relocated nor protected.

Status Register 1 (SR1)
SR1 is implemented on some PDP-11 computers to provide additional capability. The LSI-11/23 does not implement this register but does respond to its bus address, 777574_8, and reads it as all 0s. This information is provided here for clarity only.

Status Register 2 (SR2)
SR2 is loaded with the 16-bit virtual address (VA) at the beginning of each instruction fetch, but is not updated if the instruction fetch fails. SR2 is read-only; a write attempt will not modify its contents. SR2 is the virtual address program counter. Upon abort, the result of SR0 bits 15, 14, or 13 being set will freeze SR2 until the SR0 abort flags are cleared. The address of SR2 is 777576_8.

Status Register 3 (SR3)
SR3 is implemented on some PDP-11 computers to provide additional capability. The LSI-11/23 implements a portion of SR3 which is reserved for future DIGITAL use. SR3 bit 4 enables I/O mapping and SR3 bit 5 enables 22-bit mapping. The address of SR3 is 772516_8. The format of SR3 is shown in the following figures.



Format of Status Register 2 (SR2)

MEMORY MANAGEMENT INSTRUCTIONS

Memory management provides communication between two spaces, as determined by the current and previous modes of the processor status word (PS). The following instructions are directly applicable to memory management.

Mnemonic	**Instruction**	**Op Code**
MFPI	move from previous instruction space	0065SS
MTPI	move to previous instruction space	0066DD
MFPD	move from previous data space	1065SS
MTPD	move to previous data space	1066DD

In the LSI-11/23, there is no distinction between instruction space and data space. The two instructions, MFPD and MTPD, execute identically to MFPI and MTPI.

CHAPTER 10
FLOATING POINT

INTRODUCTION

The floating point instruction set is a microcode option, KEF11-A, for use with the LSI-11/23. The floating point is completely software-compatible with the FP11-A used on the PDP-11/34, the FP11-E used on the PDP-11/60 and the FP11-C used on the PDP-11/70. Both single- and double-precision floating point capability are available with other features including floating-to-integer and integer-to-floating conversion.

The KEF11A microcode resides in two MOS/LSI chips contained in one 40-pin hybrid package. The FP11 requires the memory management chip, in addition to the base MOS/LSI chips, since all the floating point accumulators and status registers reside in the memory management unit.

FLOATING POINT DATA FORMATS

Mathematically, a floating point number may be defined as having the form (2**K) f, where K is an integer and f is a fraction. For a nonvanishing number, K and f are uniquely determined by imposing the condition $\frac{1}{2} \leq f < 1$. The fractional part (f) of the number is then said to be normalized. For the number 0, f must be assigned the value 0, and the value of K is indeterminate.

The KEF11A floating point data formats are derived from this mathematical representation for floating point numbers. Two types of floating point data are provided. In single precision, or floating mode, the data is 32 bits long. In double precision, or double mode, the data is 64 bits long. Sign magnitude notation is used.

Nonvanishing Floating Point Numbers

The fractional part (f) is assumed to be normalized, so that its most significant bit must be 1. This 1 is the **hidden** bit, it is not stored explicitly in the data word, but the microcode restores it before carrying out arithmetic operations. The floating and double modes respectively reserve 23 and 55 bits for f. These bits, with the hidden bit, imply effective word lengths of 24 bits and 56 bits.

Eight bits are reserved for storage of the exponent K in excess 128 (200_8) notation (i.e., K + 200_8), giving a biased exponent. Thus, exponents from −128 to +127 are represented by 0 to 377_8, or 0 to 255_{10}. For reasons listed below, a biased exponent of 0 (true exponent of -200_8), is reserved for floating point 0. Thus, exponents are restricted to the range −127 to +127 inclusive (-177_8 to $+177_8$) or, in excess 200_8 notation, 1 to 377_8.

The remaining bit of the floating point word is the sign bit. The number is negative if the sign bit is a 1.

Floating Point Zero

Because of the hidden bit, the fractional part is not available to distinguish between 0 and nonvanishing numbers whose fractional part is exactly ½. Therefore, the KEF11-A reserves a biased exponent of 0 for this purpose and any floating point number with a biased exponent of 0 either traps or is treated as if it were an exact 0 in arithmetic operations. An exact or clean 0 is represented by a word whose bits are all 0s. A dirty 0 is a floating point number with a biased exponent of 0 and a nonzero fractional part. An arithmetic operation for which the resulting true exponent exceeds 277_8 is regarded as producing a floating overflow; if the true exponent is less than -177_8, the operation is regarded as producing a floating underflow. A biased exponent of 0 can thus arise from arithmetic operations as a special case of overflow (true exponent = -200_8). Only eight bits are reserved for the biased exponent. The fractional part of results obtained from such overflow and underflow is correct.

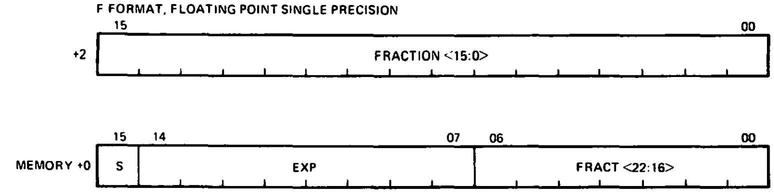
The Undefined Variable

The undefined variable is defined as any bit pattern with a sign bit of 1 and a biased exponent of 0. The term **undefined variable** is used, for historical reasons, to indicate that these bit patterns are not assigned a corresponding floating point arithmetic value. Note that the undefined variable is frequently referred to as −0 elsewhere in this specification.

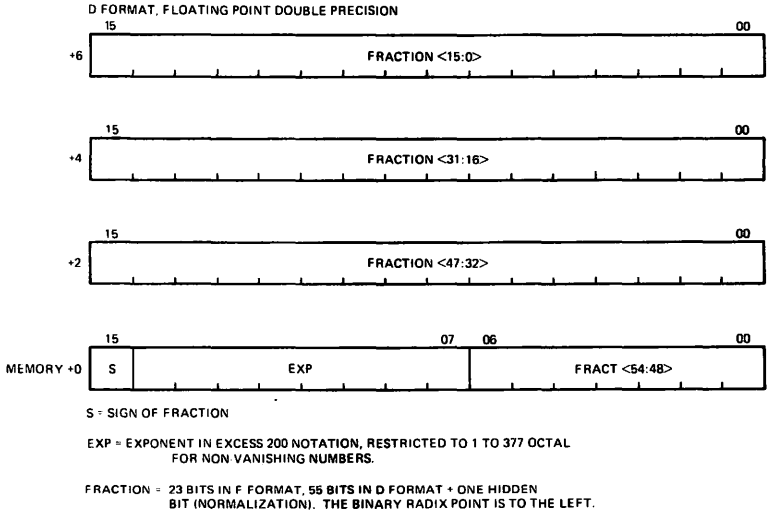
A design objective of the KEF11-A was to assure that the undefined variable would not be stored as the result of any floating point operation in a program run with the overflow and underflow interrupts disabled. This objective is achieved by storing an exact 0 on overflow and underflow, if the corresponding interrupt is disabled. This feature, together with an ability to detect reference to the undefined variable implemented by the FIUV bit mentioned later, is intended to provide the user with a debugging aid. If the presence of −0 occurs, it did not result from a previous floating point arithmetic instruction.

Floating Point Data

Floating point data is stored in words of memory as illustrated in the accompanying figures.



Single Precision Format


D FORMAT, FLOATING POINT DOUBLE PRECISION
15
00
+6
FRACTION <15:0>
15
00
+4
FRACTION <31:16>
15
00
+2
FRACTION <47:32>
15
07
06
00
MEMORY +0
S
EXP
FRACT <54:48>
S = SIGN OF FRACTION
EXP = EXPONENT IN EXCESS 200 NOTATION, RESTRICTED TO 1 TO 377 OCTAL FOR NON-VANISHING NUMBERS.
FRACTION = 23 BITS IN F FORMAT, 55 BITS IN D FORMAT + ONE HIDDEN BIT (NORMALIZATION). THE BINARY RADIX POINT IS TO THE LEFT.

Double Precision Format

The KEF11-A provides for conversion of floating point to integer format and vice-versa. The processor recognizes single precision integer (I) and double precision integer long (L) numbers, which are stored in standard 2's complement form.

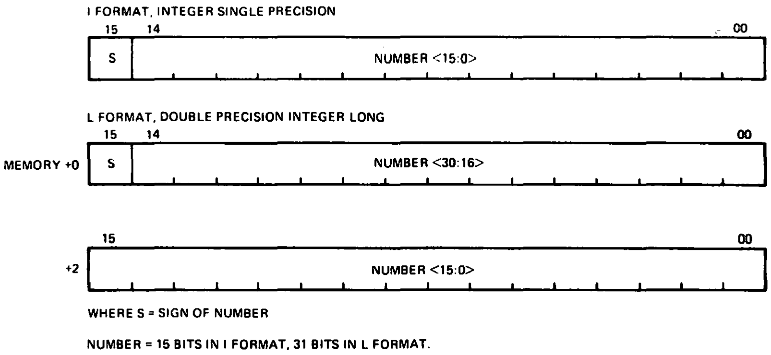
FLOATING POINT STATUS REGISTER (FPS)

This register provides mode and interrupt control for the floating point unit and conditions resulting from the execution of the previous instruction.

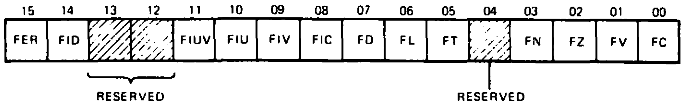
For the purposes of discussion a set bit = 1, and a reset bit = 0. Three bits of the FPS register control the modes of operation.

- Single/Double: floating point numbers can be either single or double precision.

- Short/Long: integer numbers can be 16 bits or 32 bits.
- Chop/Round: the result of a floating point operation can be either chopped or rounded. The term "chop" is used instead of "truncate" in order to avoid confusion with truncation of series used in approximations for function subroutines.


I FORMAT, INTEGER SINGLE PRECISION
15
14
00
S
NUMBER <15:0>
L FORMAT, DOUBLE PRECISION INTEGER LONG
MEMORY +0
NUMBER <30:16>
+2
NUMBER <15:0>
WHERE S = SIGN OF NUMBER
NUMBER = 15 BITS IN I FORMAT, 31 BITS IN L FORMAT.

2's Complement Format



Floating Point Status Register

The FPS register contains an error flag and four condition codes (5 bits): carry, overflow, zero, and negative, which are analogous to the processor status condition codes.

The KEF11-A recognizes six floating point exceptions:

- Detection of the undefined variable in memory
- Floating overflow
- Floating underflow
- Failure of floating-to-integer conversion
- Attempt to divide by 0
- Illegal floating opcode.

For the first four of these exceptions, bits in the FPS register enable and disable interrupts. An interrupt on either of the last two exceptions can be disabled only by setting a bit which disables interrupts on all of the exceptions as a group.

Of the thirteen FPS bits, five are set by the FPS as part of the output of a floating point instruction, the error flag (1) and condition codes (4). Any of the mode and interrupt control bits may be set by the user; the LDFPS instruction is available for this purpose. These thirteen bits are stored in the FPS register. The FPS register bits are described in the following table.

FPS Register Bits

Bit Name	**Description**
15 Floating Error (FER)	The FER bit is set by the FPP if: Division by zero occurs Illegal opcode occurs Any of the remaining occurs and the corresponding interrupt is enabled. This action is independent of the FID bit status. Also note that the KEF11-A never resets the FER bit. Once the FER bit is set by the KEF11-A, it can be cleared only by an LDFPS instruction (the RESET instruction does not clear the FER bit). This means that the FER bit is up-to-date only if the most recent floating point instruction produced a floating point exception.
14 Interrupt Disable (FID)	If the FID is set, all floating point interrupts are disabled. The FID bit is primarily a maintenance feature. It should normally be clear. In particular, it must be clear if one wishes to assure that storage of −0 by the KEF11-A is always accompied by an interrupt. Throughout the rest of this chapter, it is assumed that the FID bit is clear in all discussions involving overflow, underflow, occurrence of −0, and integer conversion errors.
13	Reserved for future DIGITAL use.
12	Reserved for future DIGITAL use.

Bit Name	FPS Register Bits Description
11 Interrupt on Undefined Variable (FIUV)	An interrupt occurs if FIUV is set and a −0 is obtained from memory as an operand of ADD, SUB, MUL, DIV, CMP, MOD, NEG, ABS, TST, or any LOAD instruction. The interrupt occurs before execution on the KEF11-A except on NEG, ABS, and TST for which it occurs after execution. When FIUV is reset, −0 can be loaded and used in any KEF11-A operation. Note that the interrupt is not activated by the presence of −0 in an AC operand of an arithmetic instruction; in particular, trap on −0 never occurs in mode 0. The LSI-11/23 will not store a result of −0 without a simultaneous interrupt.
10 Interrupt on Underflow (FIU)	When the FIU bit is set, floating underflow will cause an interrupt. The fractional part of the result of the operation causing the interrupt will be correct. The biased exponent will be too large by 400_8, except for the special case of 0, which is correct. An exception is discussed later in the detailed description of the LDEXP instruction. If the FIU bit is reset and if underflow occurs, no interrupt occurs and the result is set to exact 0.
9 Interrupt on Overflow (FIV)	When the FIV bit is set, floating overflow will cause an interrupt. The fractional part of the result of the operation causing the overflow will be correct. The biased exponent will be too small by 400_8. If the FIV is reset and overflow occurs, there is no interrupt. The KEF11-A returns exact 0. Special cases of overflow are discussed in the detailed descriptions of the MOD and LDEXP instruction.

Bit Name	**FPS Register Bits Description**
8 Interrupt on Integer Conversion Error (FIC)	When the FIC bit is set and conversion to integer instruction fails, an interrupt will occur. If the interrupt occurs, the destination is set to 0, and all other registers are left untouched. If the FIC bit is reset, the result of the operation will be the same as detailed above, but no interrupt will occur. The conversion instruction fails if it generates an integer with more bits than can fit in the short or long integer word specified by the FL bit (bit 7).
7 Floating Double Precision Mode (FD)	The FD bit determines the precision that is used for floating point calculations. When set, double precision is assumed; when reset, single precision is used.
6 Floating Long Integer Mode (FL)	The FL bit is active in conversion between integer and floating point format. When set, the integer format assumed is double precision 2's complement (i.e., 32 bits). When reset, the integer format is assumed to be single precision 2's complement (i.e., 16 bits).
5 Floating Chop Mode (FT)	When the FT bit is set, the result of any arithmetic operation is chopped (or truncated). When reset, the result is rounded.
4	Reserved for future DIGITAL use.
3 Floating Negative (FN)	FN is set if the result of the last floating point operation was negative, otherwise it is reset.
2 Floating Zero (FZ)	FZ is set if the result of the last floating point operation was 0, otherwise it is reset.
1 Floating Overflow (FV)	FV is set if the last floating point operation resulted in an exponent overflow, otherwise it is reset.
0 Floating Carry (FC)	FC is set if the last operation resulted in a carry of the most significant bit. This can only occur in floating or double to integer conversion.

FLOATING EXCEPTION CODE AND ADDRESS REGISTERS

One interrupt vector is assigned to take care of all floating point exceptions (location 244_8). The six possible errors are coded in the 4-bit floating exception code (FEC) register as follows:

2	Floating opcode error
4	Floating divide by 0
6	Floating to integer conversion error
8	Floating overflow
10	Floating underflow
12	Floating undefined variable.

The address of the instruction producing the exception is stored in the floating exception address (FEA) register.

The FEC and FEA registers are updated only when one of the following occurs:

- divide by 0
- illegal opcode
- any of the other four exceptions with the corresponding interrupt enabled.

This implies that when the FER bit is set by the KEF11-A, the FEC and FEA registers are updated. If one of the last four exceptions occurs with the corresponding interrupt disabled, the FEC and FEA are not updated.

Inhibition of interrupts by the FID bit does not inhibit updating of the FEC and FEA, if an exception occurs.

The FEC and FEA do not get updated if no exception occurs. This means that the STST (store status) instruction will return current information only if the most recent floating point instruction produced an exception.

Unlike the FPS register, no instructions are provided for storage into the FEC and FEA registers.

FLOATING POINT PROCESSOR INSTRUCTION ADDRESSING

Floating point processor instructions use the same type of addressing as the central processor instructions. A source or destination operand is specified by designating one of eight addressing modes and one of eight central processor general registers to be used in the specified mode. The modes of addressing are the same as those of the central processor except mode 0. In mode 0 the operand is located in the designated floating point processor accumulator, rather than in a central processor general register. The modes of addressing are as follows:

0 = FP11 accumulator
1 = Deferred
2 = Autoincrement
3 = Autoincrement deferred
4 = Autodecrement
5 = Autodecrement deferred
6 = Indexed
7 = Indexed deferred

Autoincrement and autodecrement operate on increments and decrements of 4 for F format and 10_8 for D format.

In mode 0, the user can make use of all six KEF11-A accumulators (AC0-AC5) as source or destination. Specifying KEF11-A accumulators AC6 or AC7 will result in an illegal opcode trap. In all other modes, which involve transfer of data to or from memory or the general registers, the user is restricted to the first four KEF11-A accumulators (AC0-AC3). When reading or writing a floating point number from or to memory, the low memory word contains the most significant word of the floating point number and the high memory word the least significant word.

ACCURACY

The descriptions of the individual instructions include the accuracy at which they operate. An instruction or operation is regarded as "exact" if the result is identical to an infinite precision calculation involving the same operands. The a priori accuracy of the operands is thus ignored. All arithmetic instructions treat an operand whose biased exponent is 0 as an exact 0 (unless FIUV is enabled and the operand is −0, in which case an interrupt occurs). For all arithmetic operations, except DIV, a 0 operand implies that the instruction is exact. The same holds for DIV if the 0 operand is the dividend. But if the divisor is 0, division is undefined and an interrupt occurs.

For nonvanishing floating point operands, the fractional part is binary normalized. It contains 24 bits or 56 bits for floating mode or double mode, respectively. For ADD, SUB, MUL, and DIV, two guard bits are necessary and sufficient for the general case to guarantee return of a chopped or rounded result identical to the corresponding infinite precision operation chopped or rounded to the specified word length. With two guard bits, a chopped result has an error bound of one least significant bit (LSB). A rounded result has an error bound of ½ LSB. These error bounds are realized by the LSI-11/23 for all instructions. Both the FP11-A and the FP11-E have an error bound greater than ½ LSB for ADD and SUB.

In this handbook, an arithmetic result is called exact if no nonvanishing bits would be lost by chopping. The first bit lost in chopping is referred to as the **rounding** bit. The value of a rounded result is related to the chopped result as follows:

- if the rounding bit is 1, the rounded result is the chopped result incremented by one LSB.
- if the rounding bit is 0, the rounded and chopped results are identical.

It follows that:

- if the result is exact, rounded value = chopped value = exact value
- if the result is not exact, its magnitude
  - is always decreased by chopping
  - is decreased by rounding if the rounding bit is 0
  - is increased by rounding if the rounding bit is 1.

Occurrence of floating point overflow and underflow is an error condition: the result of the calculation cannot be correctly stored because the exponent is too large to fit into the eight bits reserved for it. However, the internal hardware has produced the correct answer. For the case of underflow, replacement of the correct answer by 0 is a reasonable resolution of the problem for many applications. This is done by the LSI-11/23 if the underflow interrupt is disabled. The error incurred by this action is an absolute rather than a relative error; it is bounded (in absolute value) by 2^{-128}. There is no such simple resolution for the case of overflow. The action taken, if the overflow interrupt is disabled, is described under FIV (bit 9).

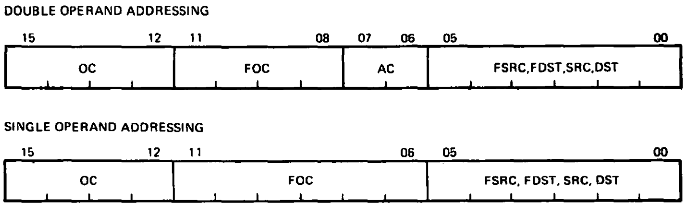
The FIV and FIU bits provide you with an opportunity to implement your own correction of an overflow or underflow condition. If such a condition occurs and the corresponding interrupt is enabled, the microcode stores the fractional part and the low eight bits of the biased exponent. The interrupt will take place and you can identify the cause by examination of the FV (floating overflow) bit or the FEC (floating exception) register. For the standard arithmetic operations ADD, SUB, MUL, and DIV, the biased exponent returned by the instruction bears the following relation to the correct exponent generated by the microcode:

- On overflow, it is too small by 400_8.
- On underflow, if the biased exponent is 0, it is correct. If it is not 0, it is too large by 400_8.

Thus, with the interrupt enable, enough information is available to determine the correct answer. You may, for example, rescale your variables (via STEXP and LDEXP) to continue a calculation. The accuracy of the fractional part is unaffected by the occurence of underflow or overflow.

FLOATING POINT INSTRUCTIONS

Each instruction that interfaces a floating point number can operate on either single or double precision numbers depending on the state of the FD mode bit. Similarly, there is a mode bit FL that determines whether a 32-bit integer (FL = 1) or a 16-bit integer (FL = 0) is used in conversion between integer and floating point representations. FSRC and FDST operands use floating point addressing modes. SRC and DST operands use CPU addressing modes.



OC = OPCODE = 17
FOC = FLOATING OPCODE
AC = FLOATING POINT ACCUMULATOR (AC0-AC3)
FSRC AND FDST USE FPP ADDRESSING MODES
SPC AND DST USE CPU ADDRESSING MODES

Floating Point Addressing Modes

Terms Used in Instruction Definitions

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 =
$2^{-128} - 2^{-127} * 1/2$

XUL = largest number that can be represented =
$2^{128} * XL$

JL = largest integer number that can be represented:
$2^{15} - 1$; FL = 0; short integer
$2^{31} - 1$; FL = 1; long integer

ABS (address) = absolute value of (address)

EXP (address) = biased exponent of (address)

.LT. = less than

.LE. = less than or equal to

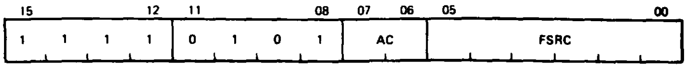
.GT. = greater than

.GE. = greater than or equal to

LSB = least significant bit

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 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.

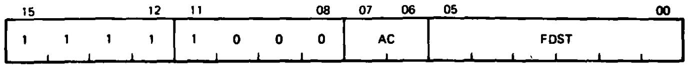
Overflow and underflow cannot occur.

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.

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 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.

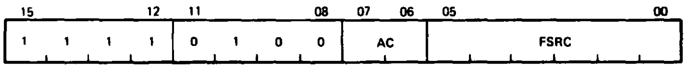
Overflow and underflow cannot occur.

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 KEF11-A. 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.

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 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 0 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.

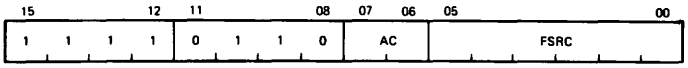
Accuracy: Errors due to overflow and underflow are described above. If neither occurs, then for oppositely signed operands with an exponent difference of 0 or 1, the answer returned is exact if a loss of significance of one or more bits can occur. Note that these are the only cases for which loss of significance of more than one bit can occur. For all other cases the result is inexact with error bounds of:

1. 1 LSB in chopping mode with either single or double precision
2. ½ LSB in rounding mode with either single or double precision.

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.

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 (AC) < 0, else FN ← 0

Description: Subtract the contents of FSRC from the contents of AC. The subtraction is carried out in single 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 0 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 like 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:

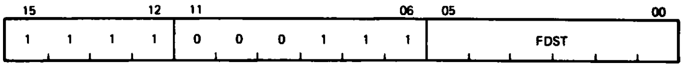
1. 1 LSB in chopping mode with either single or double precision

2. ½ LSB in rounding mode with either single or double precision.

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 (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 or double precision number, store result in same location (FDST).

Interrupts: If FIUV is enabled, trap on −0 occurs after execution.

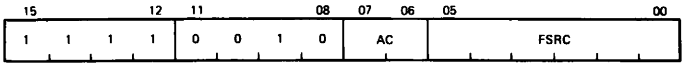
Overflow and underflow cannot occur.

Accuracy: These instructions are exact.

Special Comment: If a −0 is present in memory and the FIUV bit is enabled, then the KEF11-A stores an exact 0 in memory. The condition codes reflect an exact 0 (FZ ← 1).

MULF/MULD

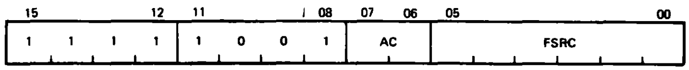
Multiply Floating/Double 171(AC)FSRC



Format:	MULF FSRC,AC
Operation:	Let PROD = (AC) * (FSRC). If underflow occurs and FIU is not enabled, AC $\leftarrow$ exact 0. If overflow occurs and FIV is not enabled, AC $\leftarrow$ exact 0. For all other cases, AC $\leftarrow$ PROD.
Condition Codes:	FC $\leftarrow$ 0 FV $\leftarrow$ 1 if overflow occurs, else FV $\leftarrow$ 0 FZ $\leftarrow$ 1 if (AC) = 0, else FZ $\leftarrow$ 0 FN $\leftarrow$ 1 if (AC) $<$ 0, else FN $\leftarrow$ 0
Description:	If the biased exponent of either operand is 0, (AC) $\leftarrow$ exact 0. For all other cases PROD is generated to 32 bits for floating mode and 64 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 0 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, the error incurred is bounded by 1 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.

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 0, it is treated as an exact 0. For FSRC this would imply division by 0; in this case the instruction is aborted, the FEC register is set to 4 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:

1. Overflow with interrupt disabled
2. Underflow with interrupt disabled.

For these exceptional cases, an exact 0 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 1 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 in enabled.

CMPF/CMPD - Use Signed Conditional Branches

Compare Floating/Double 173(AC+4)FSRC

15			12	11			08	07 06	05 00
1	1	1	1	0	1	1	1	AC	FSRC

Format: CMPF FSRC,AC

Operation: (FSRC) $\overset{-}{\leftarrow}$ (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. FSRC and the accumulator are left unchanged except as noted below.

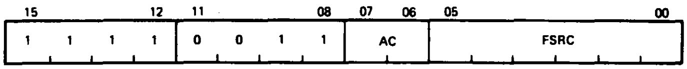
Interrupts: If FIUV is enabled, trap on −0 occurs before execution.

Accuracy: These instructions are exact.

Special Comment: An operand which has a biased exponent of 0 is treated as if it were an exact 0. In this case where both operands are 0, the FPP will store an exact 0 in AC. (i.e. Accumulator would be changed in this case).

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 .LE. f .LT. 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 .LE. g .LT. 1.

Both N and f ^{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 f 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 ACvl 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, ACvl ← 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 ACvl.

If FIV is not enabled, ACvl ← exact 0, AC ← exact 0, and −0 will never be stored.

2. If 2**L .LE. ABS(PROD) and no overflow, ACvl ← N, chopped to L bits, AC ← exact 0

   The sign and EXP of N are correct, but low-order bit information, such as parity, is lost.
3. If 1 .LE. ABS(PROD) .LT. 2**L, ACvl ← 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 1 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 eight because only 64 bits of PROD are generated.

   If 2**p .LE. ABS(N) .LT. 2**(p**1), with p >7, the low order p−7 bits of g may be in error.
4. If ABS(PROD) .LT. 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 1 LSB in chopping mode and ½ LSB in rounding 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 trigonometic function.

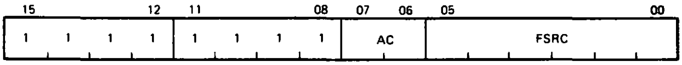
   ARG * 2/PI = N + g. The low two bits of N identify the quadrant, and g is the argument reduced to the first quadrant. The accuracy of N+g is limited to L bits because of the factor 2/PI. The accuracy of the reduced argument thus depends on the size of N.

3. To evaluate the exponential function e**x, obtain x * (log e base 2) = N + g, then e**x = (2** N) * (e**(g*1n 2)).

   The reduced argument is g * 1n2 < 1 and the factor 2**N is an exact power of 2, 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 2). The accuracy of the reduced argument thus depends on the size of N.

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.

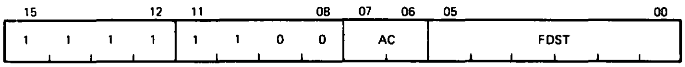
A trap occurs if FIV is enabled, and if rounding with LDCDF causes overflow. AC ← overflowed result. This result must be +0 or −0.

Underflow cannot occur.

Accuracy: LDCFD is an exact instruction. Except for overflow, described above, LDCDF incurs an error bounded by 1 LSB in chopping mode and by ½ LSB in rounding mode.

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 stored in FDST.

Interrupts: Trap on −0 will not occur even if FIUV is enabled because FSRC is an accumulator.

Underflow cannot occur.

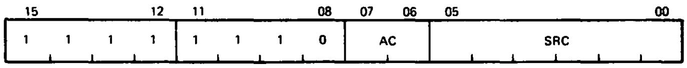
Overflow cannot occur for STCFD.

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.

LDCIF/LDCID/LDCLF/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 loaded with 0s before conversion.

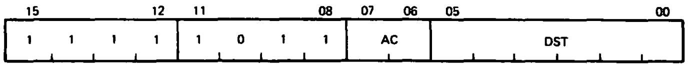
In the case of LDCLF, the fractional part of the floating point representation is chopped or rounded to 24 bits according to the state of FT (1 = chop, 0 = round).

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 1 LSB in chopping mode and by ½ LSB in rounding mode.

STCFI/STCFL/STCDI/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.

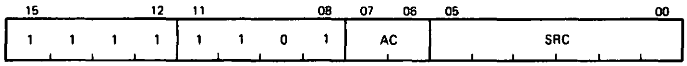
Interrupts: These instructions do not interrupt if FIUV is enabled, because the −0, if present, is in AC, not in memory.

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.

LDEXP

Load Exponent 176(AC+4)SRC



Format: LDEXP SRC,AR

Operation: If $-200_8 < SRC < 200_8$, EXP(AC) $\leftarrow$ SRC $+200_8$ and the rest of AC is unchanged.

If (SRC) > 177_8 and FIV is enabled, EXP(AC) $\leftarrow$[(SRC) + 200_8 + <7:0>.]

If (SRC) > 177_8 and FIV is disabled, AC $\leftarrow$ exact 0.

If <SRC) < -177_8 and FIU is enabled, EXP(AC) $\leftarrow$ [(SRC) + 200_8 + <7:0>.]

If (SRC) < -177_8 and FIU is disabled, AC $\leftarrow$ exact 0.

Condition Codes:
FC $\leftarrow$ 0
FV $\leftarrow$ 1 if (SRC) > 177_8, else FV $\leftarrow$ 0
FZ $\leftarrow$ 1 if (AC) = 0, else FZ $\leftarrow$ 0
FN $\leftarrow$ 1 if (AC) < 0, else FN $\leftarrow$ 0

Description: Change AC so that its unbiased exponent = (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) LE 177_8.

If SRC > 177_8, the result is treated as overflow. If SRC < -177_8, the result is treated as underflow. Note that the KEF11-A does not treat these abnormal conditions as the FP11-C and FP11-B do, but it does treat them as the FP11-A and FP11-E do.

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.

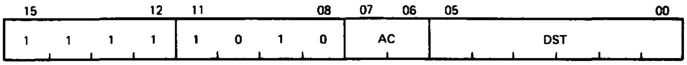
If SRC < -177_8 and FIU is enabled, trap on underflow will occur.

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 0 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 .LE. ABS(f) .LT. 1.

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, FX ← 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.

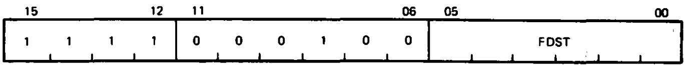
Interrupts: This instruction will not trap on −0.

Overflow and underflow cannot occur.

Accuracy: This instruction is always exact.

CLRF/CLRD

Clear Floating/Double 1704 FDST



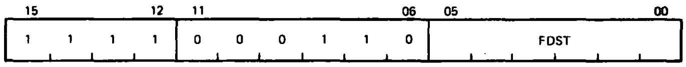
Format:	CLRF FDST
Operation:	(FDST) ← exact 0

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.

ABSF/ABSD

Make Absolute Floating/Double 1706 FDST



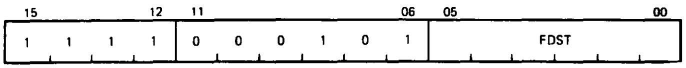
Format:	ABSF FDST
Operation:	If (FDST) < 0, (FDST) ← −(FDST). If EXP(FDST) = 0, (FDST) ← exact 0. For all other cases, (FDST) ← (FDST).

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 0 is stored in memory. The condition codes reflect an exact 0 (FZ ← 1).

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.

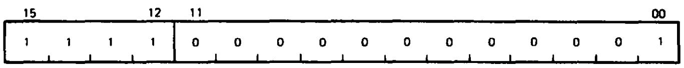
Interrupts: If FIUV is set, trap on −0 occurs after execution.

Overflow and underflow cannot occur.

Accuracy: These instructions are exact.

SETF

Set Floating Mode 170001



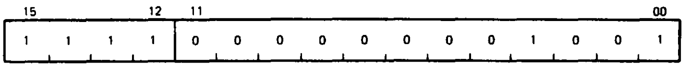
Format: SETF

Operation: FD ← 0

Description: Set the KEF11-A in single precision mode.

SETD

Set Floating Double Mode 170011



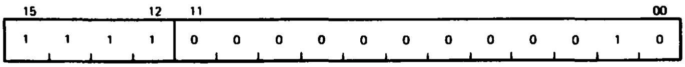
Format: SETD

Operation: FD ← 1

Description: Set the KEF11-A in double precision mode.

SETI

Set Integer Mode 177002



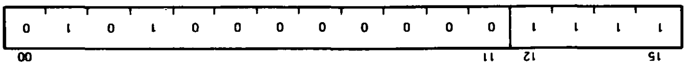
Format: SETI

Operation: FL ← 0

Description: Set the KEF11-A for short integer data.

SETL

Set Long Integer Mode 177012



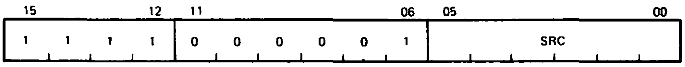
Format: SETL

Operation: FL ← 1

Description: Set the KEF11-A for long integer data.

LDFPS

Load KEF11-A's Program Status 1701 SRC



Format: LDFPS SRC

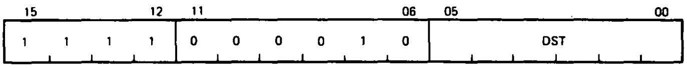
Operation: FPS ← (SRC)

Description: Load KEF11-A's 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.

STFPS

Store KEF11-A's Program Status 1702 DST



Format: STFPS DST

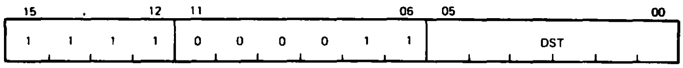
Operation: (DST) ← FPS

Description: Store KEF11-A's status register in DST.

Special Comment: Bits 13, 12, and 4 are loaded with 0. All other bits are the corresponding bits in the FPS.

STST

Store KEF11-A's Status 1703 DST



Format: STST DST

Operation: (DST) ← FEC
(DST + 2) ← FEA

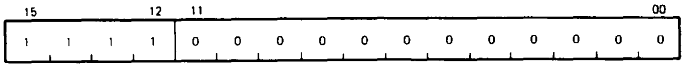
Description: Store the FEC and FEA in DST and DST+2.

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.

CFCC

Copy Floating Condition Codes 170000



Format:	CFCC

Operation:	C ← FC
	V ← FV
	Z ← FZ
	N ← FN

Description:	Copy the FP11 condition codes into the CPU's condition codes.

EXTENDED ARITHMETIC OPTION

The Extended Arithmetic Chip is an option for LSI-11 and LSI-11/2 processor modules. The option, KEV11, allows extended manipulation of fixed point numbers (fixed point arithmetic) and enables direct operations on single precision 32-bit words (floating point arithemetic).

FIXED POINT ARITHMETIC (EIS)

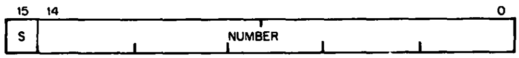
The following instructions apply to fixed point numbers:

Mnemonic	Instruction	Op Code
MUL	Multiply	070RSS
DIV	Divide	071RSS
ASH	Shift Arithmetically	072RSS
ASCH	Arithmetic Shift Combined	073RSS

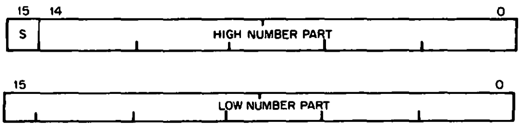
EIS

Operand formats are:

16-bit single word:



32-bit double word:



S is the sign bit.	S = 0 for positive quantities
	S = 1 for negative quantities; number is in 2's complement notation

When executing an EIS instruction, the LSI-11 and LSI-11/2 processors fetch the source operand via the DATIO bus cycle, rather than the DATI bus cycle. If the source operand is contained in a PROM or ROM location, a bus error (timeout) will occur because the processor will attempt to write into the addressed location after fetching the operand.

EIS

MUL

Multiply 070RSS

Operation: R, Rv1 ← R x(src)

Condition Codes:
N: set if product is <0; cleared otherwise
Z: set if product is 0; cleared otherwise
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 two'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. Assembler syntax is: MUL S,R.
(Note that the actual destination is R,Rv1 which reduces to just R when R is odd.)

Example: 16-bit product (R is odd)

```
CLC              ;Clear carry condition code
MOV #400,R1
MUL #10,R1
BCS ERROR        ;Carry will be set if
                 ;product is less than
                 ;-2^15 or greater than or
                 equal to 2^15
                 ;no significance lost
```

Even register is Hi-order product
Odd register is Lo-order product

Before	After
(R1)=0004000	(R1)=004000

Assembler format for all EIS instructions is:
OPR src, R

EIS

DIV

Divide 071RSS

Operation: R, Rv1 ← R, Rv1 /(src)

Condition Codes:
N: set if quotient <0; cleared otherwise
Z: set if quotient = 0; cleared otherwise
V: set if source = 0 or if the absolute value of the register is equal to or larger than the absolute value of the source. (In this case the instruction is aborted because the quotient would exceed 15 bits.)
C: set if divide by 0 attempted; cleared otherwise

Description: The 32-bit two's complement integer in R and Rv1 is divided by the source operand. The quotient is left in R; the reminder in Rv1. Division will be performed so that the remainder is of the same sign as the dividend. R must be even.

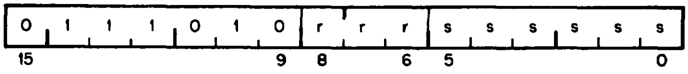
Example:
CLR R0
MOV #200001,R1
DIV #2,R0

Before	After	
(R0) = 000000	(R0) = 010000	Quotient
(R1) = 020001	(R1) = 000001	Remainder

ASH

EIS

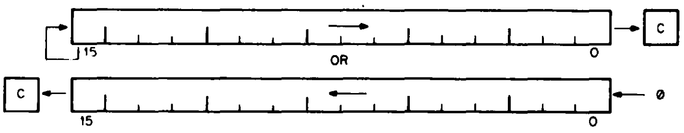
Shift Arithmetically 072RSS



Operation: R← R shifted arithmetically NN places to right or left
Where NN = low order 6 bits of source

Condition Codes:
N: set if result <0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if sign of register changed during shift; cleared otherwise
C: loaded from last bit shifted out of register

Description: The contents of the register are shifted right or left the number of times specified by the shift count. The shift count is taken as the low order 6 bits of the source operand. This number ranges from −32 to +31. Negative is a right shift and positive is a left shift.



6 LSB of source	Action in general register
011111	shift left 31 places
000001	shift left 1 place
111111	shift right 1 place
100000	shift right 32 places

Example: ASH R0, R3

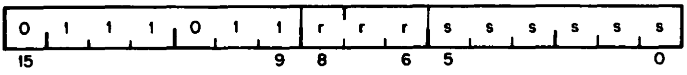
N3. #, Loc.

Before	After
(R3)=001234	(R3)=012340
(R0)=000003	(R0)=000003

EIS

ASHC

Arithmetic Shift Combined 073RSS



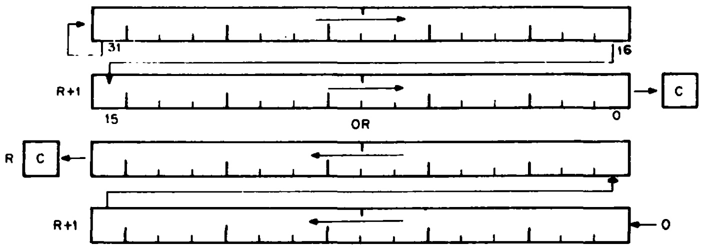
Operation: R, Rv1←R, Rv1. The double word is shifted NN places to the right or left, where NN = low order six bits of source.

Condition Codes:
N: set if result <0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if sign bit changes during the shift; cleared otherwise
C: loaded with high order bit when left shift; loaded with low order bit when right shift (loaded with the last bit shifted out of the 32-bit operand)

Description: The contents of the register and the register ORed with one are treated as one 32-bit word, R + 1 (bits 0-15) and R (bits 16-31) are shifted right or left the number of times specified by the shift count. The shift count is taken as the low order 6 bits of the source operand. This number ranges from −32 to +31. Negative is a right shift and positive is a left shift. When the register chosen is an odd number the

NOTE:
Left Shift
HI R1 → RØ LO
Right Shift
LO RØ → R1 HI

register and the register OR'ed with one are the same. In this case the right shift becomes a rotate (for up to a shift of 16). The 16 bit word is rotated right the number of bits specified by the shift count.



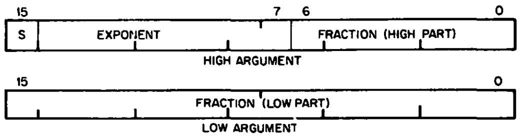
FLOATING POINT ARITHMETIC (FIS)

The Floating Point instructions used are unique to the LSI-11 and LSI-11/23 . However, the Op Codes used do not conflict with any other instructions.

Mnemonic	Instruction	Op Code
FADD	Floating Add	07500R
FSUB	Floating Subtract	07501R
FMUL	Floating Multiply	07502R
FDIV	Floating Divide	07503R

FIS

The operand format is:



S = sign of fraction; 0 for positive, 1 for negative
Exponent = 8 bits for the exponent, in excess $(200)_8$notation
Fraction = 23 bits plus 1 hidden bit (all numbers are assumed to be normalized)

The number format is essentially a sign and magnitude representation.
The format is identical with the 11/45 for single precision numbers.

Fraction

The binary radix point is to the left (in front of bit 6 of the high argument), so that the value of the fraction is always less than 1 in magnitude. Normalization would always cause the first bit after the radix point to be a 1, so that the fractional value would be between ½ and 1.This bit can be understood and not be represented directly, to achieve an extra 1 bit of resolution.

The first bit to the right of the radix point (hidden bit) is always a 1. The next bit for the fraction is taken from bit 6 of the High Argument. The result of a Floating Point operation is always rounded away from zero, increasing the absolute value of the number.

Exponent
The 8-bit exponent field (bits 14 to 7) allow exponent values between −128 and +127. Since an excess 200_8 or 128_{10} number system is used, the correspondence between actual values and coded representation is as follows:

Actual Value	**Representation**	
Decimal	**Octal**	**Binary**
+127	377	11 111 111
+1	201	10 000 001
0	200	10 000 000
−1	177	01 111 111
−128	000	00 000 000

If the actual value of the exponent is equal to −128, meaning a total value (including the fraction) or less than 2^{-128}, a trap will occur (refer to Trap section).

Example of a Number

$$+(12)_{10} = +(1100)_2$$

$$= +(2^4)_{10} \times (.11)_2 \qquad [16 \times (\tfrac{1}{2} + \tfrac{1}{4}) = 12]$$

$$\text{representation: } \overset{\text{S}}{0} \quad \overbrace{10\ 000\ 100}^{\text{Exponent}} \uparrow \ \uparrow \overbrace{1000000\ \ 0000000000000000}^{\text{Fraction}}$$

hidden bit is a 1

radix point is understood

Registers
There are no preassigned registers for the Floating Point option. A general purpose register is used as a pointer to specify a stack address. The contents of the register are used to locate the operands and answer for the Floating Point operations as follows:

(R) = high B argument address
(R)+2 = low B argument address
(R)+4 = high A argument address
(R)+6 = low A argument address

After the Floating Point operation, the answer is stored on the stack as follows:

(R)+4 = address for high part of answer
(R)+6 = address for low part of answer

where (R) is the original contents of the general register used.

After execution of the instruction, the general registers will point to the high answer, at (R)+4.

Condition Codes
Condition codes are set or cleared as shown in the Instruction Description, in the next part of this section. If a trap occurs as a function of a Floating Instruction, the condition codes are reinterpreted as follows:

V = 1, if an error occurs
N = 1, if underfow or divide by zero
C = 1, if divide by zero
Z = 0

	V	N	C	Z
Overflow	1	0	0	0
Underflow	1	1	0	0
Divide by 0	1	1	1	0

Traps
Traps will occur through vector address 244. Traps will occur due to overflow, underflow, or divide by zero conditions.

Following a trap, the general register will be unaltered, as will (R), (R) + 2, (R) +4, and (R) +6.

The condition codes in the PS that caused a trap to 244 will be set in the PS that was used while the FIS instruction was being executed. Following the trap, this PS will be pushed onto the stack. The stack must be examined following a trap to retrieve the PS and determine the reason for the trap.

Interrupts
A Floating Point instruction will be aborted if an interrupt request is issued before the instruction is near completion. The Program Counter will point to the aborted Floating Point instruction so that the interrupt will look transparent.

Assembler format is: OPR R

FIS **FADD**

Floating Add 07500R

Operation: [(R)+4, (R)+6] ← [(R)+4, (R)+6]+[(R), (R)+2]

Condition Codes:
N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: cleared

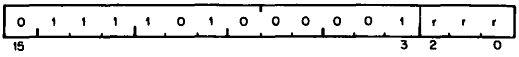
Description: Adds the A argument to the B argument and stores the result in the A argument position on the stack. General register R is used as the stack pointer for the operation.

A ← A + B

FIS **FSUB**

PUSH A
PUSH B

Floating Subtract FSUB = A-B 07501R



Operation: [(R)+4, (R)+6]←[(R)+4, (R)+6]−[(R), (R)+2]

Condition Codes:
N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: cleared

Description: subtracts the B argument from the A argument and stores the result in the A argument position on the stack.

A ← A − B

FMUL FIS

Floating Multiply **07502R**

15												3	2		0
0	1	1	1	1	0	1	0	0	0	0	1	0	r	r	r

Operation: [(R)+4, (R)+6]←[(R)+4, (R)+6]×[(R), (R)+2]

Condition Codes:
N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: cleared

Description: Multiplies the A argument by the B argument and stores the result in the A argument position on the stack.

A←A × B

(refer to note in FDIV)

FDIV FIS

PUSH Dividend
PUSH Divisor
FDIV R

Floating Divide **07503R**

15												3	2		0
0	1	1	1	1	0	1	0	0	0	0	1	1	r	r	r

Operation: [(R)+4, (R)+6]←[(R)+4, (R)+6] / [(R), (R)+2]

Condition Codes:
N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: cleared

Description: Divides the A argument by the B argument and stores the result in the A argument position on the stack. If the divisor (B argument) is equal to zero, the stack is left untouched.

A←A/B

Note: The LSI-11 processors push one word onto the stack during execution of the FMUL and FDIV instructions and pops the word from the stack when completed. Thus, the SP (R6) must point to a read/write memory location; otherwise, a bus error (timeout) will occur.

CHAPTER 11

PROGRAMMING TECHNIQUES

The LSI-11 and the PDP-11 processors offer you a great deal of programming flexiblity and power. The combination of the instruction set, the addressing modes, and the programming techniques make it possible to develop new software or to utilize old programs effectively. The programming techniques in this chapter show methods which exploit the unique capabilities of the 11 processors. The techniques specifically discussed are: position-independent coding (PIC), stacks, subroutines, interrupts, reentrancy, coroutines, recursion, processor traps, and conversion.

POSITION-INDEPENDENT CODE

The output of a MACRO-11 assembly is a relocatable object module. The task builder or linker binds one or more modules together to create an executable task image. Once built, a task can generally be loaded and executed only at the address specified at link time. This is because the linker has had to modify some instructions to reflect the memory locations in which the program is to run. Such a body of code is considered position dependent (i.e., dependent on the virtual addresses to which it was bound).

All 11 processors offer addressing modes that make it possible to write instructions that are not dependent on the virtual addresses to which they are bound. A body of such code is termed position-independent and can be loaded and executed at any virtual address. Position-independent code can improve system efficiency, both in the use of virtual address space and in the conservation of physical memory.

In multiprogramming systems like IAS and RSX-11M, it is important that many tasks be able to share a single physical copy of common code; for example, a library routine. To make the optimum use of a task's virtual address space, shared code should be position-independent. Code that is not position independent can also be shared, but it must appear in the same locations in every task using it. This restricts the placement of such code by the task builder and can result in the loss of virtual addressing space.

The construction of position-independent code is closely linked to the proper use of 11 addressing modes. The remainder of this explanation assumes that you are familiar with the addressing modes described in Chapter 7.

All addressing modes involving only register references are position independent. These modes are:

R	register mode
(R)	register deferred mode
(R)+	autoincrement mode
@(R)+	autoincrement deferred mode
−(R)	autodecrement mode
@−(R)	autodecrement deferred mode

When using these addressing modes, you are guaranteed position independence, providing that the contents of the registers have been supplied independent of a particular memory location.

The relative addressing modes are position-independent when a relocatable address is referenced from a relocatable instruction. These modes are as follows:

A	PC relative mode
@A	PC relative deferred mode

Relative modes are not position-independent when an absolute address (that is, a non-relocatable address) is referenced from a relocatable instruction. In this case, absolute addressing (i.e., @#A) may be employed to make the reference position-independent.

Index modes can be either position-independent or position dependent, according to their use in the program. These modes are as follows:

X(R)	index mode
@X(R)	index deferred mode

If the base, X, is an absolute value (e.g., a control block offset), the reference is position-independent. For example:

```
        MOV     2(SP),R0            ;POSITION INDEPENDENT
N=4
        MOV     N(SP),R0            ;POSITION INDEPENDENT
```

If, however, X is a relocatable address, the reference is position-dependent. For example:

```
        CLR     ADDR(R1)            ;POSITION DEPENDENT
```

Immediate mode can be either position independent or not, according to its use. Immediate mode references are formatted as follows:

#N	immediate mode

When an absolute expression defines the value of N, the code is position-independent. When a relocatable expression defines N, the code is position-independent. That is, immediate mode references are position independent only when N is an absolute value.

Absolute mode addressing is position-independent only in those cases where an absolute virtual location is being referenced. Absolute mode addressing references are formatted as follows:

```
            @#A        absolute mode
```

An example of a position-independent absolute reference is a reference to the directive status word ($DSW) from a relocatable instruction. For example:

```
        MOV     @#$DSW,R0         ;RETRIEVE DIRECTIVE
                                  ;STATUS
```

EXAMPLES

The RSX-11 library routine, PWRUP, is a FORTRAN callable subroutine to establish or remove a user power failure AST (Asynchronous System Trap) entry point address. Imbedded within the routine is the actual AST entry point which saves all registers, effects a call to the user-specified entry point, restores all registers on return, and executes an AST exit directive. The following examples are excerpts from this routine. The first example has been modified to illustrate position-dependent references. The second example is the position-independent version.

Position-Dependent Code

```
PWRUP:
        CLR     -(SP)             ;ASSUME SUCCESS
        CALL    .X.PAA            ;PUSH (SAVE)
                                  ;ARGUMENT ADDRESSES
                                  ;ONTO STACK
        .WORD   1.,$DSW           ;CLEAR DSW, AND
                                  ;SET R1=R2=SP
        MOV     $OTSV,R4          ;GET OTS IMPURE
                                  ;AREA POINTER
        MOV     (SP)+,R2          ;GET AST ENTRY
                                  ;POINT ADDRESS
        BNE     10$               ;IF NONE SPECIFIED,
                                  ;SPECIFY NO POWER
        CLR     -(SP)             ;RECOVERY AST SERVICE
        BR      20$               ;
```

```
10$:                                    ;
        MOV     R2,F.PF(R4)             ;SET AST ENTRY POINT
        MOV     #BA,-(SP)               ;PUSH AST SERVICE
                                        ;ADDRESS
20$:                                    ;
        CALL    .X.EXT                  ;ISSUE DIRECTIVE, EXIT.
        .BYTE   109.,2.                 ;
        .
        .
        .
BA:     MOV     R0,-(SP)                ;PUSH (SAVE) R0
        MOV     R1,-(SP)                ;PUSH (SAVE) R1
        MOV     R2,-(SP)                ;PUSH (SAVE) R2
```

Position-Independent Code

```
PWRUP:
        CLR     -(SP)                   ;ASSUME SUCCESS
        CALL    .X.PAA                  ;PUSH ARGUMENT
                                        ;ADDRESSES ONTO
                                        ;STACK
        .WORD   1.,$DSW                 ;CLEAR DSW, AND
                                        ;SET R1=R2=SP.
        MOV     @#$OTSV,R4              ;GET OTS IMPURE
                                        ;AREA POINTER
        MOV     (SP)+,R2                ;GET AST ENTRY
                                        ;POINT ADDRESS
        BNE     10$                     ;IF NONE SPECIFIED,
                                        ;SPECIFY NO POWER
        CLR     -(SP)                   ;RECOVERY AST SERVICE
        BR      20$
10$:                                    ;
        MOV     R2,F.PF(R4)             ;SET AST ENTRY POINT
        MOV     PC,-(SP)                ;PUSH CURRENT LOCATION
        ADD     #BA-.,(SP)              ;COMPUTE ACTUAL
                                        ;LOCATION
                                        ;OF AST
20$:
        CALL    .X.EXT                  ;ISSUE DIRECTIVE, EXIT.
        .BYTE   109.,2.
;
;ACTUAL AST SERVICE ROUTINE:
;
;       1)SAVE REGISTERS
;       2)EFFECT A CALL TO SPECIFIED SUBROUTINE
;          3)RESTORE REGISTERS
;          4)ISSUE AST EXIT DIRECTIVE
;
BA:        MOV     R0,-(SP)          ;PUSH (SAVE) R0
           MOV     R1,-(SP)          ;PUSH (SAVE) R1
           MOV     R2,-(SP)          ;PUSH (SAVE) R2
```

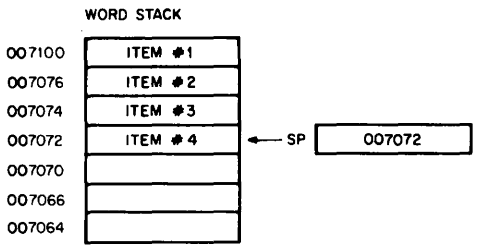
The position-dependent version of the subroutine contains a relative reference to an absolute symbol ($OTSV) and a literal reference to a relocatable symbol (BA). Both references are bound by the task builder to fixed memory locations. Therefore, the routine will not execute properly as part of a resident library if its location in virtual memory is not the same as the location specified at link time.

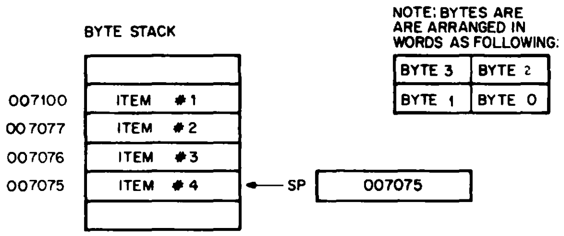
In the position-independent version, the reference to $OTSV has been changed to an absolute reference. In addition, the necessary code has been added to compute the virtual location of BA based upon the value of the program counter. In this case, the value is obtained by adding the value of the program counter to the fixed displacement between the current location and the specified symbol. Thus, execution of the modified routine is not affected by its location in the image's virtual address space.

STACKS

The stack is part of the basic design architecture of the 11 processors. It is an area of memory set aside by the programmer or by the operating system for temporary storage and linkage. It is handled on a LIFO (last-in/first-out) basis, where items are retrieved in the reverse of the order in which they were stored. On an 11 processor, a stack starts at the highest location reserved for it and expands linearly downward to a lower address as items are added to the stack.

You do not need to keep track of the actual locations into which data is being stacked. This is done automatically through a stack pointer. To keep track of the last item added to the stack, a general register always contains the memory address when the last item is stored in the stack. Any register except register 7 (the PC) may be used as a stack pointer under program control; however, instructions associated with subroutine linkage and interrupt service automatically use register 6 as a *hardware* stack pointer. For this reason, R6 is frequently referred to as the system SP. Stacks may be maintained in either full word or byte units. This is true for a stack pointed to by any register except R6, which must be organized in full word units only. Byte stacks require instructions capable of operating on bytes rather than full words.





Items are added to a stack using the autodecrement addressing mode. Adding items to the stack is called PUSHing, and is accomplished by the following instructions:

```
MOV     Source,-(SP)         ;MOV Contents of Source Word
                             ;onto the stack
                           or
MOVB    Source,-(SP)         ;MOVB Source Byte onto
                             ;the stack
```

Data is thus PUSHed onto the stack.

Removing data from the stack is called a POP (popping from the stack). This operation is accomplished using the autoincrement mode:

```
MOV      (SP)+, Destination       ;MOV Destination Word
                                  ;off the stack
                           or
MOVB     (SP)+, Destination       ;MOVB Destination Byte
                                  ;off the stack
```

After an item has been popped, its stack location is considered free and available for other use. The stack pointer points to the last used location, implying that the next lower location is free. Thus, a stack may represent a pool of temporary storage locations.

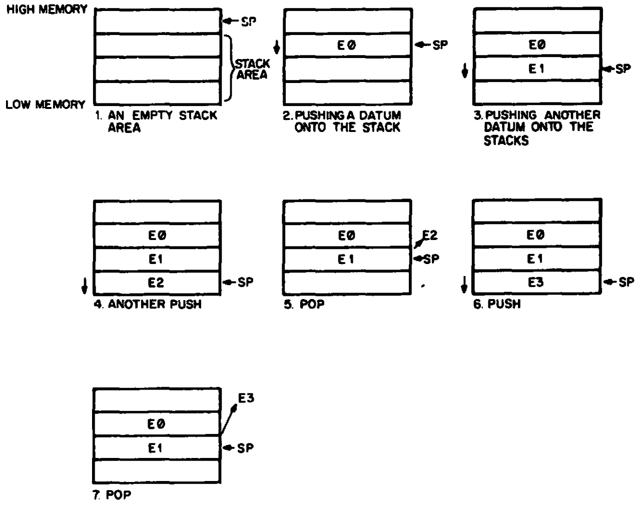


Illustration of Push and Pop Operations

Uses for the stack

- Often one of the general purpose registers must be used in a subroutine or interrupt service routine and then returned to its original value. The stack can be used to store the contents of the registers involved.
- The stack is used in storing linkage information between a subroutine and its calling program. The JSR instruction, used in calling a subroutine, requires the specification of a linkage register along with the entry address of the subroutine. The content of this linkage register is stored on the stack, so as not to be lost, and the return address is moved from the PC to the linkage register. This provides a pointer back to the calling program so that successive arguments may be transmitted easily to the subroutine.
- If no arguments need be passed by stacking them after the JSR instruction, the PC may be used as the linkage register. In this case, the result of the JSR is to move the return address in the calling program from the PC onto the stack and replace it with the entry address of the called subroutine.
- In many cases, the operations performed by the subroutine can be applied directly to the data located on or pointed to by a stack without the need ever actually to move the data into the subroutine area.

```
;CALLING PROGRAM
MOV     SP,R1           ;R1 IS USED AS THE STACK
JSR     PC,SUBR         ;POINTER HERE

;SUBROUTINE
ADD     (R1)+,(R1)      ;ADD ITEM #1 to #2,PLACE
                        ;RESULT IN ITEM #2,
                        ;R1 POINTS TO
                        ;ITEM #2 NOW
```

Because the hardware already uses general purpose register R6 to point to a stack for saving and restoring PC and processor status word (PS) information, it is convenient to use this same stack to save and restore immediate results and to transmit arguments to and from subroutines. Using R6 in this manner permits extreme flexibility in nesting subroutines and interrupt service routines.

Since arguments may be obtained from the stack by using some form of register indexed addressing, it is sometimes useful to save a temporary copy of R6 in some other register which has been saved at the beginning of a subroutine. If R6 is saved in R5 at the beginning of the subroutine, R5 may be used to index the arguments while R6 is free to be incremented and decremented in the course of being used as a stack pointer. If R6 had been used directly as the base for indexing and not "copied," it might be difficult to keep track of the position in the argument list, since the base of the stack would change with every autoincrement/decrement which occurs.

However, if the contents of R6 (SP) are saved in R5 before any arguments are pushed onto the stack, the position relative to R5 would remain constant.

Return from a subroutine also involves the stack, as the return instruction, RTS, must retrieve information stored there by the JSR.

When a subroutine returns, it is necessary to "clean up" the stack by eliminating or skipping over the subroutine arguments. One way this can be done is by insisting that the subroutine keep the number of arguments as its first stack item. Returns from subroutines then involve calculating the amount by which to reset the stack pointer, resetting the stack pointer, then storing the original contents of the register used as the copy of the stack pointer.

- Stack storage is used in trap and interrupt linkage. The program counter and the processor status word of the executing program are pushed on the stack.

- When using the system stack, nesting of subroutines, interrupts, and traps to any level can occur until the stack overflows its legal limits.
- The stack method is also available for temporary storage of any kind of data. It may be used as a LIFO list for storing inputs, intermediate results, etc.

As an example of stack use consider this situation: a subroutine (SUBR) wants to use registers 1 and 2, but these registers must be returned to the calling program with their contents unchanged. The subroutine could be written as follows:

```
                                  Assembler
Address         Octal Code        Syntax          Comments

076322          010167 SUBR:      MOV R1,TEMP1    ;save R1
076324          000074            *
076326          010267            MOV R2,TEMP2    ;save R2
076330          000072            *
.               .                 .
.               .                 .
.               .                 .
076410          016701            MOV TEMP1,R1    ;restore R1
076412          000006            *
076414          0167902           MOV TEMP2,R2    ;restore R2
076416          000004            *
076420          000297            RTS PC
076422          000000            TEMP1: 0
076424          000000            TEMP2: 0
```

* Index Constants

OR: Using the Stack

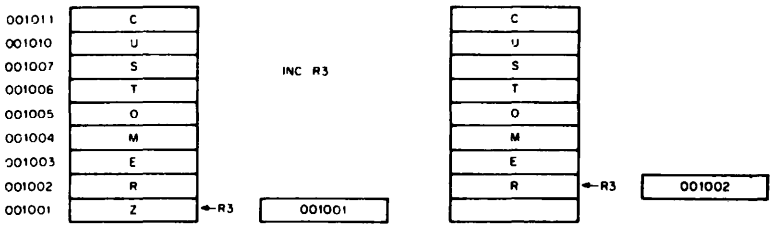
R3 has been previously set to point to the end of an unused block of memory.

```
                                  Assembler
Address         Octal Code        Syntax          Comments

010020          010143 SUBR:      MOV R1,-(R3)    ;push R1
010022          010243            MOV R2,-(R3)    ;push R2
.               .                 .
.               .                 .
.               .                 .
.               .                 .
010130          012302            MOV (R3)+,R2    ;pop R2
010132          012301            MOV (R3)+,R1    ;pop R1
010134          000207            RTS PC
```

Note: In this case R3 was used as a stack pointer.

The second routine uses four fewer words of instruction code and two words of temporary "stack" storage. Another routine could use the same stack space at some later point. Thus, the ability to share temporary storage in the form of a stack is a way to save on memory use.

As another example of stack use, consider the task of managing an input buffer from a terminal. As characters come in, you may wish to delete characters from the line; this is accomplished very easily by maintaining a byte stack containing the input characters. Whenever a backspace is received, a character is "popped" off the stack and eliminated from consideration. In this example, you have the choice of "popping" characters to be eliminated by using either the MOVB (MOVE BYTE) or INC (INCREMENT) instructions.



Byte Stack Used as a Character Buffer

NOTE that in this case the increment instruction (INC) is preferable to MOVB, since it accomplishes the task of eliminating the unwanted character from the stack by readjusting the stack pointer without the need for a destination location. Also, the stack pointer (SP) used in this example cannot be the system stack pointer (R6) because R6 may point only to word (even) locations.

DELETING ITEMS FROM A STACK
To delete one item:

INC SP or TSTB(SP)+ for a byte stack

To delete two items:

ADD#2,SP or TST(SP)+ for word stack

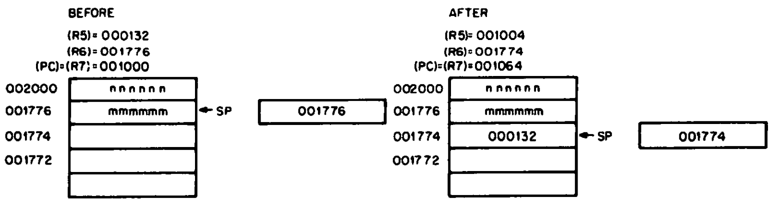
To delete fifty items from a word stack:

ADD #100.,SP

SUBROUTINE LINKAGE

The contents of the linkage register are saved on the system stack when a JSR is executed. The effect is the same as if a MOV reg,−(R6) had been performed. Following the JSR instruction, the same register is loaded with the memory address (the contents of the current PC), and a jump is made to the entry location specified.

The JSR figure gives the before and after conditions when executing the subroutine instructions JSR R5,1064.



JSR

Because the 11 hardware already uses general purpose register R6 to point to a stack for saving and restoring PC and PS (processor status word) information, it is convenient to use this same stack to save and restore intermediate results and to transmit arguments to and from subroutines. Using R6 this way permits nesting subroutines and interrupt service routines.

Return from a Subroutine

An RTS instruction provides for a return from the subroutine to the calling program. The RTS instruction must specify the same register as the one the JSR instruction used in the subroutine call. When the RTS is executed, the register specified is moved to the PC, and the top of the stack to be placed in the register specified. Thus, an RTS PC has the effect of returning to the address specified on the top of the stack.

Subroutine Advantages

There are several advantages to the PDP-11 subroutine calling procedure, affected by the JSR instruction.

- Arguments can be passed quickly between the calling program and the subroutine.
- If there are no arguments, or the arguments are in a general register or on the stack, the JSR PC,DST mode can be used so that none of the general purpose registers are used for linkage.

- Many JSRs can be executed without the need to provide any saving procedure for the linkage information, since all linkage information is automatically pushed onto the stack in sequential order. Returns can be made by automatically popping this information from the stack in the order opposite to the JSRs.

Such linkage address bookkeeping is called automatic "nesting" of subroutine calls. This feature enables you to construct fast, efficient linkages in a simple, flexible manner. It also permits a routine to call itself in those cases where this is meaningful.

INTERRUPTS

An interrupt is similar to a subroutine call, except that it is initiated by the hardware rather than by the software. An interrupt can occur after the execution of an instruction.

Interrupt-driven techniques are used to reduce CPU waiting time. In direct program data transfer, the CPU loops to check the state of the DONE/READY flag (bit 7) in the peripheral interface. Using interrupts, the system actually ignores the peripheral, running its own low-priority program until the peripheral initiates service by setting the DONE bit. The interrupt enable bit in the control status register must have been set at some prior point. The CPU completes the instruction being executed and then is interrupted and vectors to an interrupt service routine. The service routine will transfer the data and may perform calculations with it. After the interrupt service routine has been completed, the computer resumes the program that was interrupted by the peripheral's high-priority request.

With interrupt service routines, linkage information is passed so that a return to the main program can be made. More information is necessary for an interrupt sequence than for a subroutine call because of the random nature of interrupts. The complete machine state of the program immediately prior to the occurrence of the interrupt must be preserved in order to return to the program without any noticeable effects. This information is stored in the processor status word (PS). Upon interrupt, the contents of the program counter (PC) (address of next instruction) and the PS are automatically pushed onto the R6 system stack. The effect is the same as if:

```
MOV PS,-(SP)        ;Push PS
MOV PC,-(SP)        ;Push PC
```

had been executed.

The new contents of the PC and PS are loaded from two preassigned consecutive memory locations which are called "vector addresses."

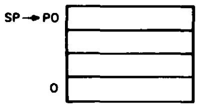
The first word contains the interrupt service routine entry address (the address of the service routine program sequence), and the second word contains the new PS which will determine the machine status, including the operational mode and register set to be used by the interrupt service routine. The contents of the vector address are set under program control.

After the interrupt service routine has been completed, an RTI (return from interrupt) is performed. The top two words of the stack are automatically "popped" and placed in the PC and PS respectively, thus resuming the interrupted program.

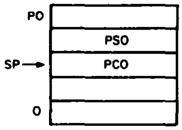
Nesting

Interrupts can be nested in much the same manner that subroutines are nested. In fact, it is possible to nest any arbitrary mixture of subroutines and interrupts without any confusion. By using the RTI and RTS instructions, respectively, the proper returns are automatic.

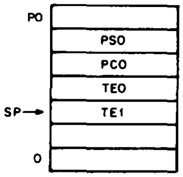
1. Process 0 is running; SP is pointing to location P0.



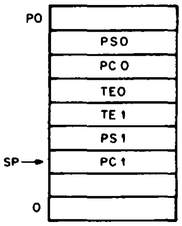
2. Interrupt stops process 0 with PC = PC0, and status = PS0; starts process 1.



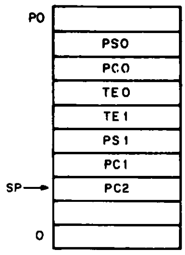
3. Process 1 uses stack for temporary storage (TE0, TE1).



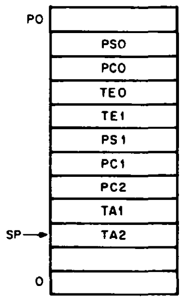
4. Process 1 interrupted with PC = PC1 and status = PS1; process 2 is started.



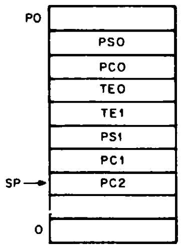
5. Process 2 is running and does a JSR R7,A to subroutine A with PC = PC2.



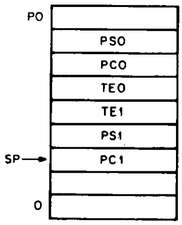
6. Subroutine A is running and uses stack for temporary storage.



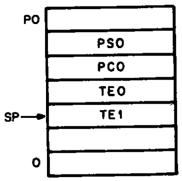
7. Subroutine A releases the temporary storage holding TA1 and TA2.



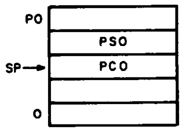
8. Subroutine A returns control to process 2 with an RTS R7; PC is reset to PC2.



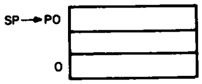
9. Process 2 completes with an RTI instructions (dismisses interrupt) PC is reset to PC(1) and status is reset to PS1; process 1 resumes.



10. Process 1 releases the temporary storage holding TE0 and TE1.



11. Process 1 completes its operation with an RTI, PC is reset to PC0, and status is reset to PS0.



Nested Interrupt Service Routines and Subroutines

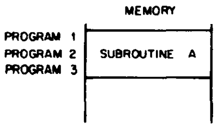
Note that the area of interrupt service programming is intimately involved with the concept of CPU and device priority levels.

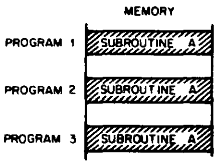
REENTRANCY

Other advantages of the 11 stack organization are obvious in programming systems that are engaged in concurrent handling of several tasks. Multi-task program environments range from simple single-user applications which manage a mixture of I/O interrupt service and background data processing, as in RT-11, to large complex multi-programming systems that manage an intricate mixture of executive and multi-user programming situations, as in RSX-11. In all these situations, using the stack as a programming technique provides flexibility and time/memory economy by allowing many tasks to use a single copy of the same routine with a simple straightforward way of keeping track of complex program linkages.

The ability to share a single copy of a program among users or among tasks is called **reentrancy**. Reentrant program routines differ from ordinary subroutines in that it is not necessary for reentrant routines to finish processing a given task before they can be used by another task. Multiple tasks can exist at any time in varying stages of completion in the same routine. Thus the following situation may occur.

(ART) (ART)





Approach

Programs 1, 2, and 3 can share Subroutine A.

Conventional Approach

A separate copy of Subroutine A must be provided for each program.

Reentrant Routines

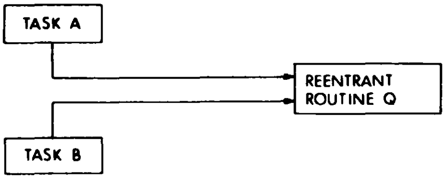
Reentrant Code

Reentrant routines must be written in pure code, code that is not self-modifying and consists entirely of instructions and constants.

Pure code (any code that consists exclusively of instructions and constants) may be used when writing any routine, even if the completed routine is not to be reenterable. The value of using pure code whenever possible is that the resulting code:

- is generally considered easier to debug
- can be kept in read-only memory (is read-only protected)

Using reentrant code, control of a routine can be shared as follows:



Sharing Control of a Routine

- Task A requests processing by Reentrant Routine Q.
- Task A temporarily relinquishes control of Reentrant Routine Q before it completes processing.
- Task B starts processing the same copy of Reentrant Routine Q.
- Task B completes processing by Reentrant Routine Q.
- Task A regains use of Reentrant Routine Q and resumes where it stopped.

Writing Reentrant Code

In an operating system environment, when one task is executing and is interrupted to allow another task to run, a context switch occurs which causes the processor status word and current contents of the general purpose registers GPRs to be saved and replaced by the appropriate values for the task being entered. Therefore, reentrant code should use the GPRs and the stack for any counters, pointers, or data that must be modified or manipulated in the routine.

The context switch occurs whenever a new task is allowed to execute. It causes all of the GPRs, the PS, and often other task-related information to be saved in an impure area, then reloads these registers and locations with the appropriate data for the task being entered. Notice that one consequence of this is that a new stack pointer value is loaded into R6, therefore causing a new area to be used as the stack when the second task is entered.

The following should be observed when writing reentrant code:

- All data should be in or pointed to by one of the general purpose registers.
- A stack can be used for temporary storage of data or pointers to impure areas within the task space. The pointer to such a stack would be stored in a GPR.
- Parameter addresses should be used by indexing and indirect reference rather than by putting them into instructions within the code.
- When temporary storage is accessed within the progam, it should be by indexed addresses, which can be set by the calling task in order to handle any possible recursion.

Use of Reentrant Code

Reentrant code is used whenever more than one task may reference the same code without requiring that each task complete processing with the code before the next may use it.

COROUTINES

In some programming situations it happens that several program segments or routines are highly interactive. Control is passed back and forth between the routines, each going through a period of suspension before being resumed. Since the routines maintain a symmetric relationship with each other, they are called **coroutines.**

Coroutines are two program sections, either subordinate to the other, which can call each other. The nature of the call is "I have processed all I can for now, so you can execute until you are ready to stop, then I will continue."

The coroutine call and return are identical, each being a jump to subroutine instruction with the destination address being on top of the stack and the PC serving as the linkage register, i.e.,

```
JSR PC,@(R6)+
```

Coroutine Calls

The coding of coroutine calls is made simple by the 11 stack feature. Initially, the entry address of the coroutine is placed on the stack and from that point the

```
JSR PC,@(R6)+
```

instruction is used for both the call and the return statements. The result of this JSR instruction is to exchange the contents of the PC and the top element of the stack, and so permit the two routines to swap control and resume operation where each was terminated by the previous swap.

For example:

Routine A	**Stack**	**Routine B**	**Comments**
.		.	LOC is pushed
.		.	onto the stack
.		.	to prepare for
MOV #LOC,-(SP)	LOC <-SP		the corou-
.			tine call.
.			

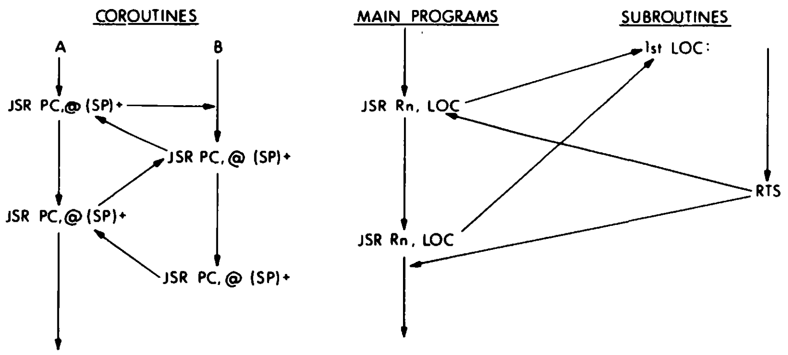
```
                                    LOC:
JSR PC,@(SP)+  PC0  <-SP                      .     When the call
(PC0)                                         .     is executed,
                                              .     the PC from
                                                    routine A is
                                                    pushed on the
                                                    stack and exe-
                                                    cution contin-
                                                    ues at LOC.
                                    JSR PC,@(SP)+   Routine B can
      .        PC1<-       SP       (PC1)           return control
      .                                       .     to routine A
      .                                       .     by another
                                              .     coroutine call.
                                                    PC0 is popped
                                                    from the stack
                                                    and execution
                                                    resumes in
                                                    routine A just
                                                    after the call
                                                    to Routine B,
                                                    i.e., at PC0.
                                                    PC1 is saved
                                                    on the stack
                                                    for a later
                                                    return to
                                                    Routine B.
```

Coroutine Example

Notice that the coroutine linkage cleans up the stack with each transfer of control.

Coroutines Versus Subroutines

- A subroutine can be considered to be subordinate to the main or calling routine, but a coroutine is considered to be on the same level, as each coroutine calls the other when it has completed current processing.
- A subroutine executes, when called, to the end of its code. When called again, the same code will execute before returning. A coroutine executes from the point after the last call of the other coroutine. Therefore, the same code will not be executed each time the coroutine is called. For example,



Coroutines vs. Subroutines

- The call and return statements for coroutines are the same:

  JSR PC,@(SP)+

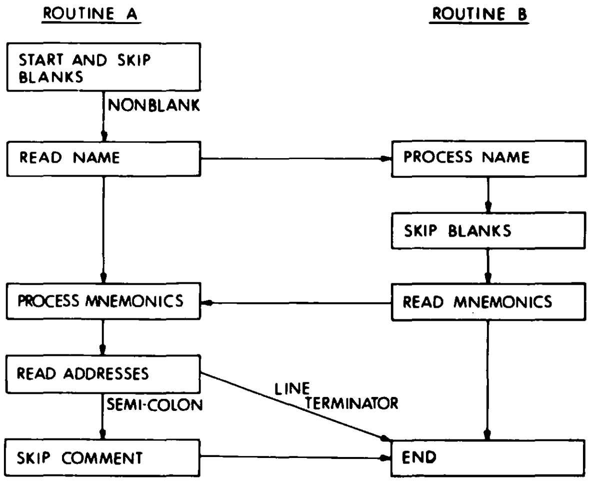
  This one instruction also cleans up the stack with each call.

  The last coroutine call will leave an address on the stack that must be popped if no further calls are to be made.
- Each coroutine call returns to the coroutine code at the point after the last exit with no need for a specific entry point label, as would be required with subroutines.

Using Coroutines
- Coroutines should be used whenever two tasks must be coordinated in their execution without obscuring the basic structure of the program. For example, in decoding a line of assembly language code, the results at any one position might indicate the next process to be entered. Where a label is detected, it must be processed. If no label is present, the operator must be located, etc.
- Coroutines should be employed to add clarity to the process being performed, to ease in the debugging phase, etc.

Examples
An assembler must perform a lexicographic scan of each assembly language statement during pass one of the assembly process. The various steps in such a scan should be separated from the main program flow to add to the program clarity and to aid in debugging by isolating many details. Subroutines would not be satisfactory here, as

too much information would have to be passed to the subroutine each time it was called. This subroutine would be too isolated. Coroutines could be effectively used here with one routine being the assembly-pass-one routine and the other extracting one item at a time from the current input line.



Coroutine Path

Coroutines can be utilized in I/O processing. The example shows coroutines used in double-buffered I/O using IOX. The flow of events might be described as:

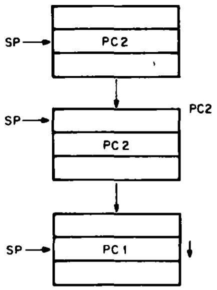
	Write 01	
	Read I1	concurrently
	Process I2	
then		
	Write 02	
	Read I2	concurrently
	Process I1	

Routine #1 is operating, it then executes:

```
        MOV #PC2,-(R6)
        JSR PC,@(R6)+
```

with the following results:

1. PC2 is popped from the stack and the SP autoincremented.
2. SP is autodecremented and the old PC (i.e. PC1) is pushed.
3. Control is transferred to the location PC2 (i.e. Routine # 2).



Routine #2 is operating, it then executes:

```
        JSR PC,@(R6)+
```

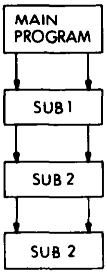
with the result that PC2 is exchanged for PC1 on the stack and control is transferred back to Routine #1.

Coroutine Interaction

RECURSION

An interesting aspect of a stack facility, other than its providing for automatic handling of nested subroutines and interrupts, is that a program may call on itself as a sub-routine just as it can call on any other routine. Each new call causes the return linkage to be placed on the stack, which, as it is a last-in/first-out queue, sets up a natural unraveling to each routine just after the point of departure.

Typical flow for a recursive routine might be something like this:



Recursive Routine Flow

The main program calls function one, SUB 1, which calls function two, SUB 2, which recurses once before returning.

Example:

```
DNCF:    ,
         ,
         ,
         BEQ 1$            ;TO EXIT RECURSIVE LOOP
         JSR R5,DNCF       ;RECURSE
1$       ,
         ,
         ,
         RTS R5            ;RETURN TO 1$ FOR
                           ;EACH CALL, THEN TO
                           ;MAIN PROGRAM
```

The routine DNCF calls itself until the variable tested becomes equal to zero, then it exits to 1$ where the RTS instruction is executed, returning to the 1$ once for each recursive call and one final time to return to the main program.

In general, recursion techniques will lead to slower programs than the corresponding interactive techniques, but the recursion will give shorter programs in memory space used. Both the brevity and clarity produced by recursion are important in assembly language programs.

Uses of Recursion

Recursion can be used in any routine in which the same process is required several times. For example, a function to be integrated may contain another function to be integrated, i.e., to solve for XM

where:

$$XM = 1 + \quad F(X)$$

and:

$$F(X) = \quad G(X)$$

Another use for a recursive function could be in calculating a factorial function because

$$FACT(N) = FACT(N-1) * N$$

Recursion should terminate when N = 1.

The macro processor within MACRO-11, for example, is itself recursive, as it can process nested macro definitions and calls. For example, within a macro definition, other macros can be called. When a macro call is encountered within definition, the processor must work recursively, i.e., to process one macro before it is finished with another, then to continue with the previous one. The stack is used for a separate storage area for the variables associated with each call to the procedure.

As long as nested definitions of macros are available, it is possible for a macro to call itself. However, unless conditionals are used to terminate this expansion, an infinite loop could be generated.

PROCESSOR TRAPS

There are a series of errors and programming conditions which will cause the central processor to trap to a set of fixed locations. These include power failure, odd addressing errors, stack errors, time out errors, memory parity errors, memory management violations, floating point processor exception traps, use of reserved instructions, use of the T bit in the processor status word, and use of the IOT, EMT, and TRAP instructions.

Power Failure

Whenever AC power drops below 95 volts for 115v power (190 volts for 230v) or outside a limit of 47 to 73 Hz, as measured by DC voltage, the power-fail sequence is initiated. The central processor automatically traps to location 24 and the power-fail program has 2 msec. to save all volatile information (data in registers), and to condition peripherals for power fail.

When power is restored, the processor traps to location 24 and executes the power-up routine to restore the machine to its state prior to power failure.

Odd Addressing Errors

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

Time-out Errors

These errors occur when a master synchronization pulse is placed on the UNIBUS and there is no slave pulse within a certain length of time. This error usually occurs in attempts to address non-existent memory or peripherals.

The offending instruction is aborted and the processor traps through location 4.

Reserved Instructions

There is a set of illegal and reserved instructions which cause the processor to trap through location 10.

Vector Address and Trap Errors

000	(reserved)
004	CPU errors
010	Illegal and reserved instructions
014	BPT, breakpoint trap
020	IOT, input/output trap
024	Powerfail
030	EMT, emulator trap
034	TRAP instruction

TRAP INSTRUCTIONS

Trap instructions provide for calls to emulators, I/O monitors, debugging packages, and user-defined interpreters. A trap is effectively an interrupt generated by software. When a trap occurs, the contents of the current program counter (PC) and program status word (PS) are pushed onto the processor stack and replaced by the contents of a 2-word trap vector containing a new PC and new PS. The return sequence from a trap involves executing an RTI or RTT instruction which restores the old PC and old PS by popping them from the stack. Trap vectors are located at permanently assigned fixed addresses.

The EMT (trap emulator) and TRAP instructions do not use the low-order byte of the word in their machine language representation. This allows user information to be transferred in the low-order byte. The new value of the PC loaded from the vector address of the TRAP or EMT instructions is typically the starting address of a routine to access and interpret this information. Such a routine is called a **trap handler.**

The trap handler must accomplish several tasks. It must save and restore all necessary GPRs, interpret the low byte of the trap instruction and call the indicated routine, serve as an interface between the calling program and this routine by handling any data that need be passed between them, and, finally, cause the return to the main routine.

Uses of Trap Handlers

The trap handler can be useful as a patching technique. Jumping out to a patch area is often difficult because a 2-word jump must be performed. However, the 1-word TRAP instruction may be used to dispatch to patch areas. A sufficient number of slots for patching should first be reserved in the dispatch table of the trap handler. The jump can then be accomplished by placing the address of the patch area into the table and inserting the proper TRAP instruction where the patch is to be made.

The trap handler can be used in a program to dispatch execution to any one of several routines. Macros may be defined to cause the proper expansion of a call to one of these routines. For example,

```
            .MACRO SUB2 ARG
            MOV ARG, R0
            TRAP +1
            .ENDM
```

When expanded, this macro sets up the one argument required by the routine in R0 and then causes the trap instruction with the number 1 in the lower byte. The trap handler should be written so that it recognizes a 1 as a call to SUB2. Notice that ARG here is being transmitted to SUB2 from the calling program. It may be data required by the routine or it may be a pointer to a longer list of arguments.

In an operating system environment like RT-11, the EMT instruction is used to call system or monitor routines from a user program. The monitor of an operating system necessarily contains coding for many functions, i.e., I/O, file manipulation, etc. This coding is made accessible to the program through a series of macro calls, which expand into EMT instructions with low bytes indicating the desired routine, or group of routines to which the desired routine belongs. Often a GPR is designated to be used to pass an identification code to further indicate to the trap handler which routine is desired. For example, the macro expansion for a resume execution command in RT-11 is as follows:

```
            .MACRO .RSUM
            CM3, 2.
            .ENDM
```

and CM3 is defined as

```
            .MACRO CM3 CHAN, CODE
            MOV #CODE *400,R0
.IIF NB     CHAN,BISB CHAN,R0
            EMT 374
            .ENDM
```

Notice the EMT low byte is 374. This is interpreted by the EMT handler to indicate a group of routines. Then the contents of R0 (high byte) are tested by the handler to identify exactly which routine within the group is being requested, in this case routine number 2. (The CM3 call of the .RSUM is set up to pass the identification code.)

Summary of PDP-11 Processor Trap Vectors:

VECTOR ADDRESS	FUNCTION SERVED
4	Illegal instructions (JSR, JMP for mode 0) Bus errors (odd address error, timeout) Stack limit (Red Zone, Yellow Zone) Illegal internal address Microbreak
10	Reserved instruction XFC with UCS disabled SPL, MTPS, MFPS FADD, FSUB, FMUL, FDIV HALT in user mode
14	Trace (T bit)
20	IOT
24	Power fail
30	EMT
34	TRAP
114	Cache parity error UNIBUS memory parity error UCS parity error
244	Floating point exception
250	Memory management (KT) abort

CONVERSION ROUTINES

Almost all assembly language programs require the translation of data or results from one form to another. Coding that performs such a transformation will be called a conversion routine in this handbook. Several commonly used conversion routines are included in the following pages.

Almost all assembly language programs involve some type of conversion routines, octal to ASCII, octal to decimal, and decimal to ASCII being a few of the most widely used.

Arithmetic multiply and divide routines are fundamental to many conversion routines.

Division is typically approached in one of two ways.

1. The division can be accomplished through a combination of rotates and subtractions.

Examples:
Assume the following code and register data; to make the example easier, also assume a 3-bit word.

```
DIV:    MOV #3,-(SP)            ;SET UP DIGIT COUNTER
        CLR -(SP)               ;CLEAR RESULT
1$      ASL (SP)
        ASL R1
        ROL R0
        CMP R0,R3
        BLT 2$
        SUB R3,R0               ;R0 CONTAINS REMAINDER
        INC (SP)                ;INCREMENT RESULT
2$      DEC 2 (SP)              ;DECREMENT COUNTER
        BNE $1
```

Therefore, to divide 7 by 2:

```
        R0=000                  remainder
        R1=111                  seven-multiplicand
        R3=010                  two-multiplier
        C bit=0

        STACK
        011                     counter
        000                     quotient
```

Following through the coding, the quotient, remainder, and dividend all shift left, manipulating the most significant digit first, etc.

At the conclusion of the routine:

```
        R0=001                  remainder
        R1=000
        R3=010

        STACK
        000                     counter
        011                     quotient
```

1. This number is here just to set up No. 2.
2. A second method of division occurs by repeated subtraction of the powers of the divisor, keeping a count of the number of subtractions at each level.

   Example:
   To divide $221_{10}$by 10, first try to subtract powers of 10 until a non-negative value is obtained, counting the number of subtractions of each power.

```
 221
-1000
```

negative so go to next lower power, count for $10^3=0$.

```
 221
-100

 121    count for 10^2=1.
-100

  21    count = 2
-100
```

negative, so reduce power.
count for $10^2=2$

```
  21
 -10

  11    count for 10^1=1.

  11
 -10

   1    count=2
 -10
```

negative, so count for $10^1=2$.

No lower power, so remainder is 1.

Answer = 022, remainder 1.

Multiplication can be done through a combination of rotates and additions or through repetitive additions.

Example:
Assume the following code and a 3-bit word.

```
CLR R0              ;HIGH HALF OF ANSWER
MOV #3,CNT          ;SET UP COUNTER
MOV R1,MULT;        ;MULTIPLICAND

MORE:               ROR R2
                    BCC NOW
                    ADD  MULT,R0  ;IF INDICATED,
ADD
                          ;MULTIPLICAND
        NOW:              ROR R0
                          ROR R1
                          DEC CNT
                          BNE MORE
        MULT:             0
        CNT:              0
```

The following conditions exist for 6 times 3:

R0 = 000 — high order half of result
R1 = 110 — multiplicand
R3 = 011 — multiplier

After the routine is executed:

R0 = 010 — high order half of result
R1 = 010 — low order half of result
R2 = 100
CNT = 0
MULT = 110

Example:
Multiplication of R0 by 50_8(101000).

```
        MUL50:            MOV R0,-(SP)
                          ASL R0
                          ASL R0
                          ADD (SP)+,R0
                          ASL R0
                          ASL R0
                          ASL R0
                          RETURN
```

If R0 contains 7:

R0 = 111

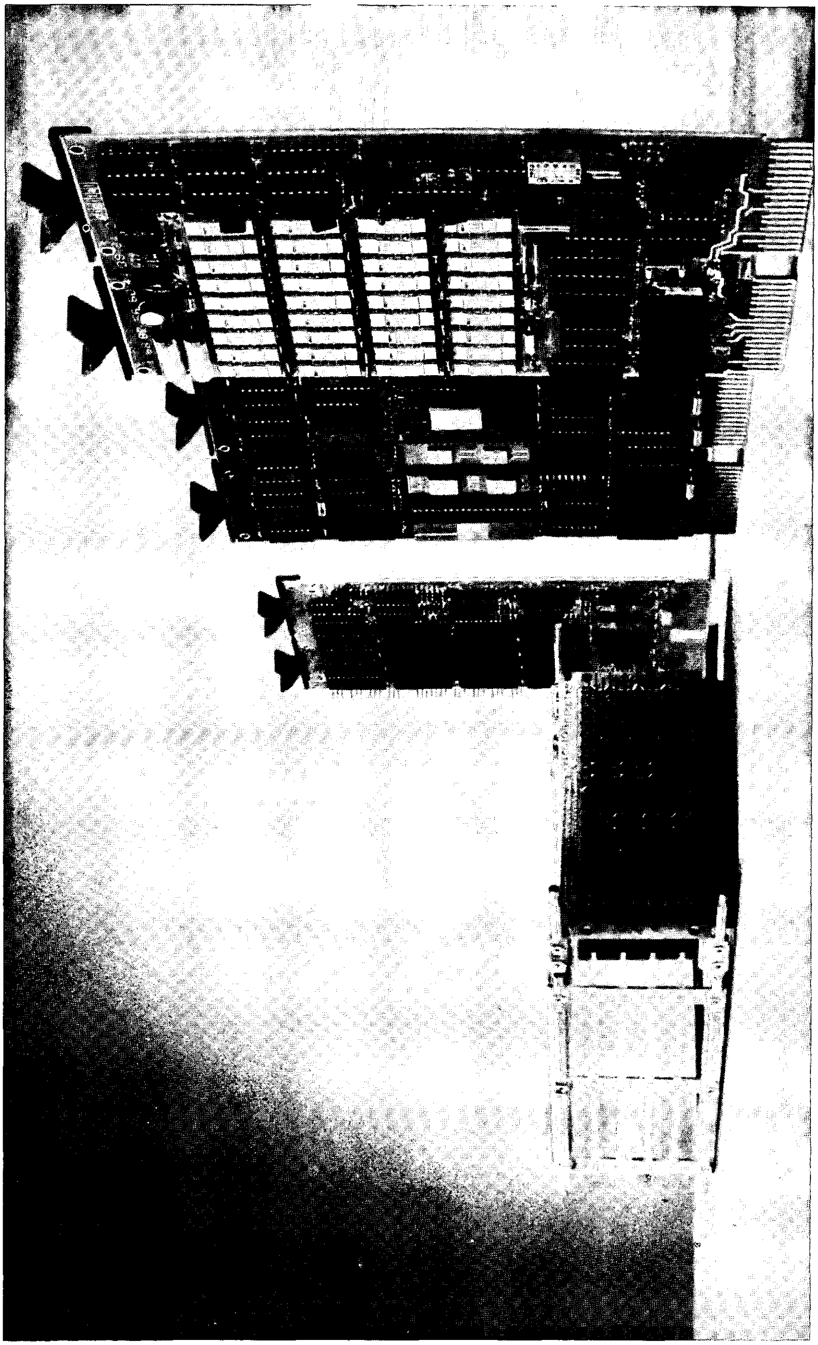
After execution;

R0 = 100011000
$(7*50_8 = 430_8)$.

ASCII CONVERSIONS

The conversion of ASCII characters to the internal representation of a number as well as the conversion of an internal number to ASCII in I/O operations presents a challenge. The following routine takes the 16-bit word in R1 and stores the corresponding six ASCII characters in the buffer addressed by R2.

```
OUT:     MOV     #5,R0            ;LOOP COUNT
LOOP:    MOV     R1,-(SP)         ;COPY WORD INTO STACK
         BIC     #177770,@SP      ;ONE OCTAL VALUE
         ADD     #'0,@SP          ;CONVERT TO ASCII
         MOVB    (SP)+,-(R2)      ;STORE IN BUFFER
         ASR     R1               ;SHIFT
         ASR     R1               ; RIGHT
         ASR     R1               ;   THREE
         DEC     R0               ;TEST IF DONE
         BNE     LOOP             ;NO, DO IT AGAIN
         BIC     #177776,R1       ;GET LAST BIT
         ADD     #'0,R1           ;CONVERT TO ASCII
         MOVB    R5,-(R2)         ;STORE IN BUFFER
         RTS     PC               ;DONE,RETURN
```



CHAPTER 12
MEMORIES

MMV11-A

MMV11-A 4K BY 16-BIT CORE MEMORY

GENERAL

The MMV11-A 4K by 16-bit core memory option provides nonvolátile read/write storage of user programs and data. Memory 4K addressing is user-selected by switches contained on the option. The MMV11-A is LSI-11 bus-compatible but it is limited to backplanes that contain the LSI-11 bus in both A/B and C/D slots. The MMV11-A is capable of either programmed I/O data transfers with the processor or transfers with another LSI-11 DMA bus device.

FEATURES

- 4096 by 16-bit capacity
- Typical access time = 425 ns (475 ns maximum); full read/restore cycle time = 1.15 μs
- Nonvolatile read/write storage – stored data remains valid when power is removed
- User-selected bank address – three switches allow the user to select the bank address for the option
- +5 V and +12 V power – only the normal backplane power is required to power the option
- No adjustments, no periodic maintenance

SPECIFICATIONS

Identification	G653/H223
Size	Two quads
Power	
Standby	5.0 Vdc ± 5% at 3.0 A 12.0 Vdc ± 3% at 0.2 A
Operational	5.0 Vdc ± 5% at 7.0 A 12.0 Vdc ± 3% at 0.6
Bus Loads	
AC	1.9
DC	1.0

MMV11-A

CONFIGURATION

General

The MMV11-A is contained on two modules which are mated to comprise a single assembly as shown in Figure 1. The modules include memory interface and timing board (module type G653) and core stack (module type H223). The G653 module includes handles and retractors on the top edge and fingers on the bottom edge which plug into the LSI-11 bus. Circuits contained on this module include interface, control and timing logic, bus receivers and drivers, the 16-bit data paths, sense amplifiers, and a +5 Vdc to −5 Vdc inverter. The H223 module is slightly smaller, includes no handles or bus fingers, and plugs onto the No. 2 (solder) side of the G653 module via special connector pins. Spacers are located between the modules to stiffen the assembly and to maintain the 2.3 cm (0.9 in) dimension. Circuits contained on the H223 module include the 4096 by 16 core stack, 12-bit address register, X and Y drives, stack charge, temperature compensation, and a series +11 V, V_{CC} switch which removes drive power when BDCOK H goes low (power fail) or BINIT L is asserted.

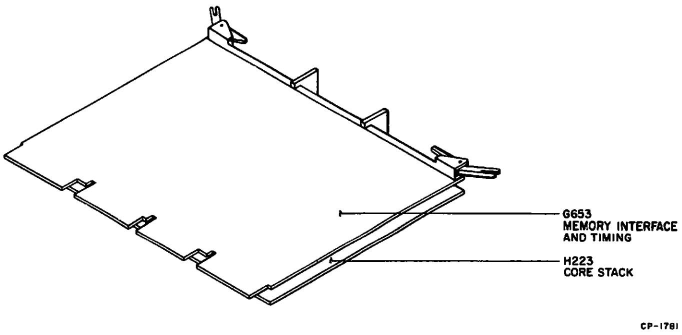


Figure MMV11-A-1
MMV11-A Core Memory Option

The MMV11-A core memory option comprises two modules (G653 and H223) that are mated by connector pins in a single assembly. It requires two device locations (electrical positions) on the backplane when installed in H9270 slots A4–D4; otherwise, because of its total thickness (2.3 cm, 0.9 in), the MMV11-A requires four physical device locations when installed in any other backplane slot. Memory capacity is 4096 16-bit words. Switches select the 4K bank address to which the MMV11-A will respond.

NOTE

The MMV11-A can only be installed in backplanes that contain the LSI-11 bus in both A/B and C/D slots.

Switch-Selected Addressing

The only preparation required for the MMV11-A before it is installed in the backplane is to select its bank address. This is accomplished by opening or closing switches in appropriate address bit locations to produce the desired bank address decoding.

MMV11-A bank address switches are located on the G653 module (component side) as shown in Figure 2. Bank address switches are used as shown in Figure 3, which illustrates a 16-bit address and how switches are assigned to each address bit. Open or close switches to produce the desired bank address as shown in Figure 3.

Backplane Jumpers

The BDMGI L and BIAKI L bus lines must be jumpered to BDMGO L and BIAKO L lines, respectively, under the H223 module when installed between the processor and I/O device interface modules in order to maintain daisy-chain signal continuity.

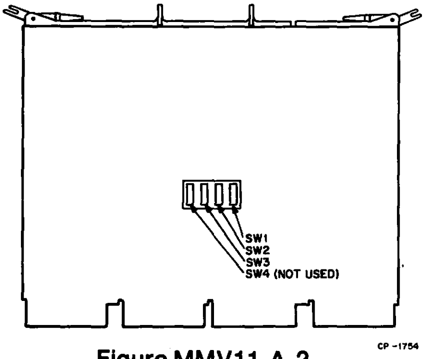
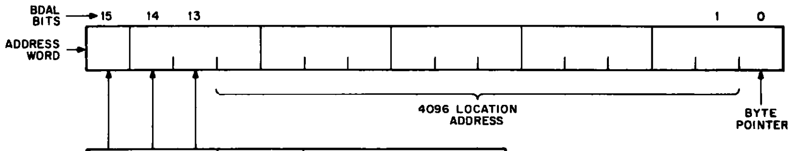


Figure MMV11-A-2
Bank Address Switch Locations



SW3	SW2	SW1	BANK	ADDRESSES
ON	ON	ON	0	0 - 17776
ON	ON	OFF	1	20000 - 37776
ON	OFF	ON	2	40000 - 57776
ON	OFF	OFF	3	60000 - 77776
OFF	ON	ON	4	100000 - 117776
OFF	ON	OFF	5	120000 - 137776
OFF	OFF	ON	6	140000 - 157776
OFF	OFF	OFF	7	160000 - 177776

NOTE:
Bank 7 is normally reserved for peripherals.

MR-0856

Figure MMV11-A-3
MMV11-A Addressing

Pins which must be connected are:

From	To	Signal
C01N2	C04M2	BIAKI/OL
C01S2	C04R2	BDMGI/OL

PROGRAMMING
The program must terminate and issue a HALT instruction within 2 ms of the time that the BPOK signal indicates that a power failure trap occurred. Failure to do so could result in the loss of data in one or more memory locations.

FUNCTIONAL DESCRIPTION

General
The MMV11-A memory is a read/write, random access, coincident current magnetic core type with a cycle time of 1.15 μs and an access time of 425 ns. It is organized in a 3D, 3-wire planar configuration. Word length is 16 bits and the memory consists of 4096 (4K) words.

Major functions contained in the MMV11-A are shown in Figure 4. Memory data can be stored (written) or read by executing appropriate bus cycles: DATO (16-bit word) write; DATOB (8-bit byte) write; DATI (16-bit word) read; DATIO (16-bit word) read-modify-write; and DATIOB (16-bit word) read-modify-(8-bit byte) write.

Each of the functions shown in Figure 4 is briefly described below.

Bus Receivers and Drivers – These devices interface directly with the LSI-11 bus and the G653 logic circuits. BDAL bus drivers are gated on by DATA OUT L during a read operation [DATI or the input portion of a DATIOB(B) bus cycle].

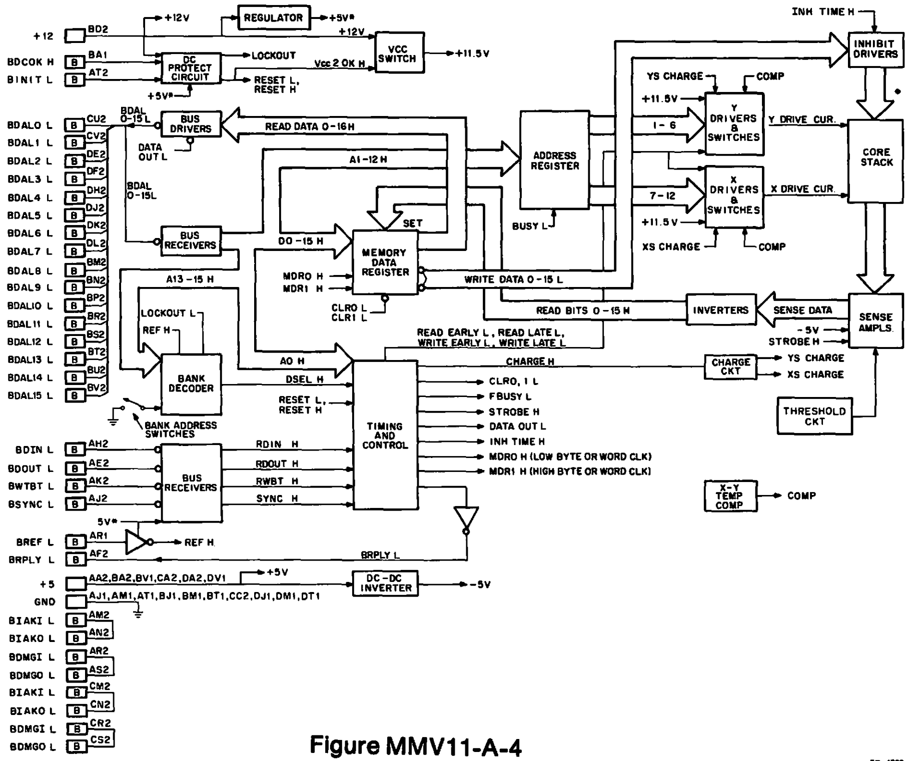


Figure MMV11-A-4
MMV11-A Logic Block Diagram

MMV11-A

Bank Decoder – The bank decoder receives address bits A13–15 L and responds when the bank address is as user-selected on the three bank address switches on the G653 module. It responds by producing an acitve DSEL H signal which initiates memory cycle timing. This signal is enabled only when power is normal and bus initialize or referesh operations are not in progress.

Timing and Control – Timing and control circuits receive bus and internal control signals and generate appropriate read/write timing and control signals. It also generates the BRPLY L signal in response to BDIN L and BDOUT L.

Address Register – The address register stores the 12-bit word address within the 4K bank during the addressing portion of the bus cycle. Latched bits LA1–6H are applied to Y drive circuits and LA7–12H are applied to X drive circuits.

X and Y Drives – X and Y drive circuits control X and Y read/write currents through all core mats. Address decoding activates 1 out of 64 X wires and 1 out of 64 Y wires. Because the active X and Y wires each have one-half the current required for core saturation, only 1 core out of 4096 cores in each core mat is saturated. Direction of current is determined by a read or write operation.

Core Stack – The core stack comprises sixteen 4096-core mats. Each mat is associated with one memory bit position at all 4096 locations. Each core has three wires passing through it: one X, one Y, and one sense/inhibit wire. The sense/inhibit wire passes through all 4096 cores in one mat. Hence, the stack contains 16 sense/inhibit lines.

The sense/inhibit line ends terminate at sense amplifier inputs. During a write operation, an inhibit current, equal to saturation current, is applied to the center of the sense/inhibit line when a logical 0 is to be written in the addressed core. This current splits and one-half saturation current flows through all cores in the mat and into termination diodes at the sense amplifier inputs. The wire is threaded through the cores in a manner that causes the current to flow in a direction opposite to that of the Y write current; this prevents core saturation, which would write a logical 1 in the addressed core.

Sense Amplifiers – Sense amplifiers respond to induced voltage impulses during the read cycle. They are strobed during a critical time of the cycle, producing an active (high) output when a logical 1 is read, regardless of the induced polarity on the two ends of the sense/inhibit wires for each bit.

Inverters – The inverters receive sense amplifier outputs, invert them, and direct-set previously cleared memory data register bits when a logical 1 is sensed.

Memory Data Register – The 16-bit memory data register is cleared upon entry to a read cycle; sensed logical 1s set appropriate bits. During a restore cycle (DATI bus cycle) (no memory contents are to be modified), the same bits (low-active) are written into the same addressed location. During a write cycle [DATO, DATOB, or the write portion of a DATIO(B) bus cycle], bus data bits are clocked into the high and/or low byte(s), depending on the type of bus cycle (word or high byte or low byte).

Inhibit Drivers – Inhibit drivers, one for each bit position, produce an inhibit current during the write cycle at INH TIME H if a logical 0 is to be written. The current inhibits core saturation, which would produce a stored logical 1.

Charge Circuit – The charge circuit applies the correct operating voltage to X and Y drive circuits during the read and write memory cycles to prevent "sneak" currents through the unselected stack diodes.

X-Y Temperature Compensation – X-Y temperature compensation circuits alter drive currents over the required operating temperature range to provide reliable operation.

DC-DC Inverter – The dc-dc inverter circuit generates −5 V power for sense amplifiers from the +5 V power.

DC Protection – DC protection circuits respond to an active BINIT L or passive BDCOK H signal by producing active LOCKOUT L, RESET H, RESET L, and passive VCC2OK H signals. These signal conditions prevent memory circuit operation and the possible loss of stored data.

V_{CC} Switch – The V_{CC} switch applies +11.5 V to X and Y driver circuits when not in an initialize or power-fail condition.

Core Addressing

When a memory location is addressed, 1 core in each of the 16 mats is accessed for a read or write operation. Figure 5 illustrates a portion of the X-Y drive and associated circuits for one Y wire. Six address bits (A1–A6) select 1 of 64 Y wires. A similar circuit (not shown) involving the remaining six address bits (A7–A12) selects 1 of 64 X wires. Hence, by placing 64 cores (in each mat) on each Y wire and passing a different X wire through each core, 1 of 64 cores on the active Y wire will be selected. Since the remaining Y wires have a similar 64 cores each and

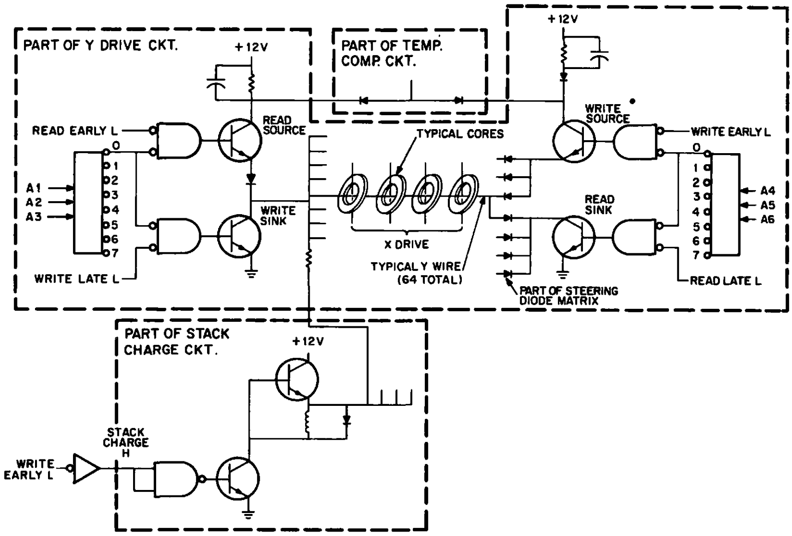


Figure MMV11-A-5
MMV11-A Core Addressing

receive the same X wires, 64 X 64, or 1 out of 4096 addressing is accomplished in each of the core mats. A single Y wire is driven as described below.

Two 1:8 (octal) decoders are used in Y wire selection, each receiving three address bits from the address register. Only one output from each decoder will be active during addressing. Assuming address XX00 (the zeros are the Y portion of the 12-bit address), the portion of the Y drive circuit shown will be enabled. During a read operation, READ EARLY L goes active and turns on one of the eight read current source transistors. A diode in its emitter circuit couples the drive to eight Y wires, each terminating at the diode steering matrix. The diodes provide a read current path to all eight read sink transistors. READ LATE L goes active 25 ns after the Y source is turned on, and turns on one of the eight read sink transistors, completing a read current path to ground. Hence, 1 of 64 Y wires is selected, producing a read half-current through 64 cores in all memory mats. Similar X drive circuits will produce an X read half-current in 64 cores in each mat in exactly the same manner. Only one core in each mat will receive an X and a Y read half-current, causing the core to saturate in the 0 state. If the core was previously in the 1 state, a voltage pulse will be induced in the sense/inhibit wire as it switches to the 0 state.

A write cycle is always preceded by a read cycle. The write operation is similar to the read operation, except write current flows through the addressed wire in a direction opposite to the read current direction. The core in each mat receiving X and Y write half-currents will respond by saturating in the 1 state. However, since a 0 may be desired, a third wire (sense/inhibit) will conduct a half-current which opposes the magnetizing effect of the Y write currents. Thus, core saturation is not attained and the cores where 0s are written remain saturated in the 0 state from the previous read cycle.

Temperature compensation is applied to driver circuits via a source current, which is inversely proportional to temperature; an increase in temperature decreases available drive current.

The stack charge circuit applies a +11 V (approximately) signal to the sink ends of all X (not shown) and Y wires during the write cycle. The level is applied during WRITE EARLY time. Since WRITE LATE L occurs 25 ns after WRITE EARLY L, the write sink transistor is cut off, and the full 11 V signal charges the stray capacitance of the X-Y lines, reducing the capacitive delay effect as the X and Y write source transistors turn on; the 11 V signal also reverse biases diodes not selected by addressing circuits, preventing sneak current. The addressed sink transistor, turned on by the active WRITE LATE L signal, provides the return path for the selected X and Y wires; only those two wires will go to approximately 0 V, causing one X and one Y diode to become forward biased, enabling the write half-currents to flow. Resistors coupling the charge voltage to write sink transistors limit the charge current through the addressed write sink transistors during the remainder of the write cycle. The circuit performs the same function for read cycles by grounding the buses and preventing sneak currents through unselected stack diodes.

Read/Write Data Path
The basic read/write data path is shown in Figure 6. Upon entering a read cycle, the memory data register is cleared by CLR0 and CLR1 L. X and Y read currents produce active sense amplifier outputs for those cores containing stored logical 1s as they are switched to the 0 states. These signals are inverted and applied to the direct-set inputs of the flip-flops comprising the memory data registers, setting the appropriate bits. During a write cycle, CLR0 L (DATOB low byte address), CLR1 L (DATOB high byte address), or both CLR0 and CLR1 L (DATO word address) clear the previously read data. The bus data is then received and clocked into the register flip-flops by CLK MDR0 H and/or CLK MDR1 H, as appropriate. Write data bits are then routed to inhibit drivers which inhibit writing 1s when write bits are 0s (high). Inhibit half-current through addressed cores prevents X-Y write half-currents from switching cores to the 1 state.

The sense/inhibit wire passes through all cores in a core mat, as shown in Figure 6. The circuit shown in the figure is repeated for each of the 16 core mats. During the read portion of a memory cycle, a logical 1 stored in the addressed core will cause an induced voltage to appear on the sense/inhibit wire as the core switches from the 1 saturation state to the 0 saturation state. If a 0 was previously written, no appreciable voltage

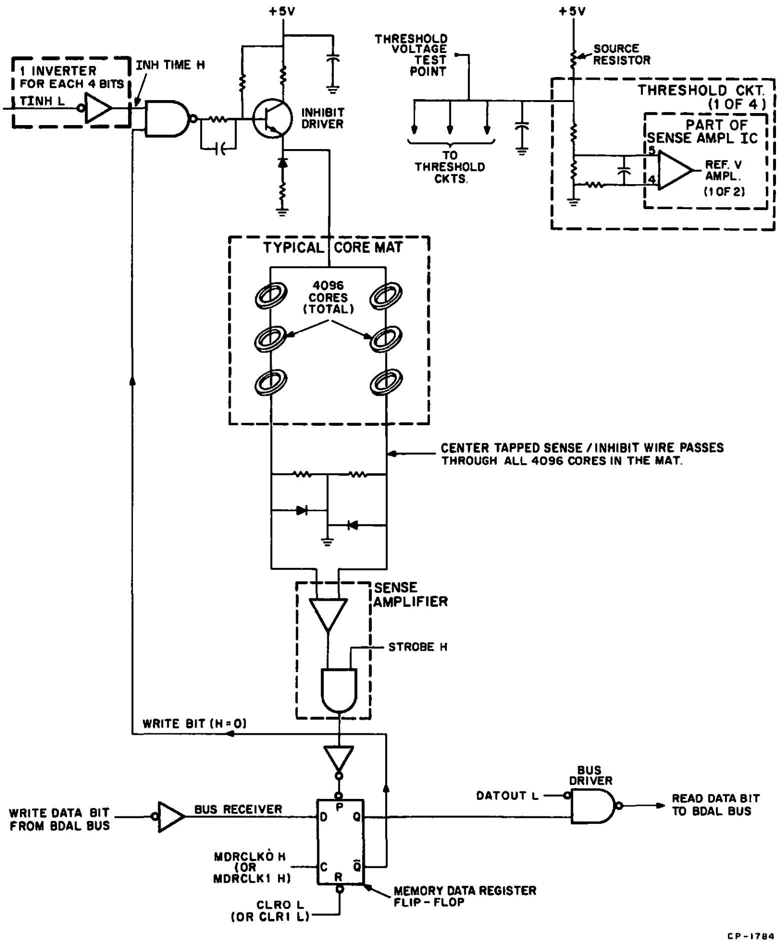


Figure MMV11-A-6
MMV11-A Read/Write Bit Data Path

is produced since the core is already saturated in the 0 state. During the read operation, the sense/inhibit wire functions as a loop whose ends terminate at the sense amplifier inputs. Any difference in potential (either polarity) will enable a sense amplifier output. STROBE H occurs during X and Y drive read currents at a critical time (the time of peak core switching output when 1s are read). Thus, only the correct voltage pulse produced when a core goes from the 1 state to the 0 state is gated into the memory data register flip-flop.

The threshold circuit establishes the signal voltage level at which a logical 1 is read during strobe time. A signal voltage magnitude greater than approximately 17 mV during strobe time results in a valid 1 level.

Signal levels less than the 17 mV threshold value are considered invalid and result in 0 levels being read. Four threshold circuits share a common source resistor. Each threshold circuit provides a reference amplifier input voltage to two sense amplifier ICs, each containing two sense amplifiers; hence, one threshold circuit provides a threshold voltage for four data bits.

When in the write portion of the memory cycle, the inhibit driver remains off if a 1 write data bit is stored in the memory data register flip-flop. However, if a 0 is to be written, the write bit is high, enabling a gate input for the inhibit driver. At INH TIME H during the write cycle, the inhibit driver produces an inhibit current equal to core saturation in a direction that would produce a logical 0. However, note that the inhibit current is applied to the center of the sense/inhibit wire. Thus, half-currents flow into each half of the sense/inhibit wire, preventing the addressed core from saturating in the 1 state. Diodes at the sense amplifier ends of the wire provide a ground return for the two inhibit half-currents. The two resistors terminate the ends of the wires. The inhibit driver transistor collector is clamped to ground through a diode and resistor to prevent breakdown during turnoff. The emitter resistor limits peak current.

Timing and Control

All memory bus cycles comprise a read and a write operation. During a DATI bus transaction, a memory read-restore cycle is executed. The data is first read and placed on the I/O bus. The same data is then written in the same addressed location. During a DATO bus transaction, a memory read-modify-write cycle is executed. After reading the contents of the addressed location, bus data is clocked into the memory data register. Previously read data is lost. The modified word is then written into the addressed location during the remainder of the cycle. If a DATOB bus transaction is being executed, only an 8-bit portion of the memory data register is modified, and one byte of the previously read word is retained for the write operation. A DATIO bus transaction actually initiates two separate memory cycles. The first cycle (read-restore) is initiated by the master device by placing the memory address on BDAL0–15 L and asserting BSYNC L. After receiving and modifying the memory read data, the master device outputs the new data to the memory and asserts BDOUT L, which initiates the next memory cycle (read-modify-write). Timing and control logic functions generate all of the timing and control signals for the memory cycles described above. Logic operation for each type of bus transaction is described in detail in the following paragraphs.

A memory cycle is initiated when the correct bank address asserted by the bus master device is decoded on the leading edge of BSYNC L. DSEL H is the decoded bank address signal; note that it is inhibited during refresh bus cycles (when BREF L is asserted), or when an initialize or power-fail condition exists. The logical state of DSEL H is clocked into the busy flip-flop on the leading edge of SYNC H (Figure 7). When DSEL H is active (high), the busy flip-flop sets and FBUSY H and FBUSY L go to their true states. FBUSY L enables one input of the read initiate gate. The remaining gate input is enabled by the negative-going pulse produced by the RC circuit connected to FBUSY L. Thus, on the leading edge of FBUS L, the state of FBUSY H is clocked into the read flip-flop, causing it to go to the set state. This sequence is shown in Figures 8 and 9.

The read-restore (DN) cycle continues as shown in Figure 8. FREAD H activates the read time generator, producing time signals prefixed with "RT." Each signal is a 225 positive-going pulse whose leading edge is delayed with respect to FREAD H. Hence, the leading edge of RT225-H, shown in Figure 7, occurs 225 ns after the leading edge of FREAD H, and approximately 275 ns after BSYNC L is asserted. Note that RT225 H is inverted and applied to the clear input of the read flip-flop, establishing the 225 ns pulse width for RT pulses. RT225 H goes low 225 ns later. This time occurs 400 ns (total) after BSYNC L is asserted and it is referenced on Figure 7 as (T500L).

The pulse produced by the RC network on the leading edge of FBUSY L is inverted to produce the CLR0 L and CLR1 L signals, which clear the memory data register for the new read data. READ EARLY occurs on the leading edge of FREAD H and remains active for the duration of RT50 H, producing a 300 ns pulse. RT25 H goes high 25 ns later, producing the READ LATE L signal; this signal remains true for the duration of RT100, resulting in a 325 ns pulse. Read data is valid at the sense amplifier inputs from 200 to 275 ns after READ EARLY L goes active. RT175 H is gated with RT50 H to produce the sense amplifier strobes STROBE 0 and 1 H. The trailing edge of RT50 H occurs 100 ns after the leading edge of RT175 H, negating the strobes. During strobe time, the sense amplifier data bits set the appropriate flip-flops that comprise the memory data register, and store the memory read data.

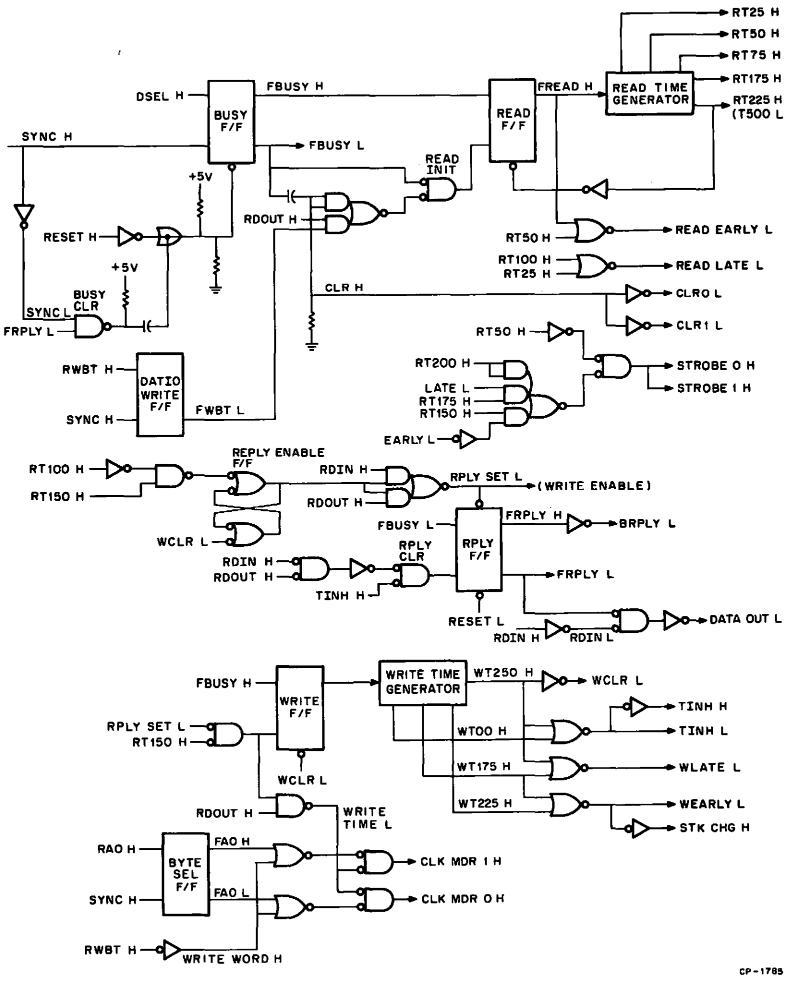


Figure MMV11-A-7
MMV11-A Timing and Control Circuits

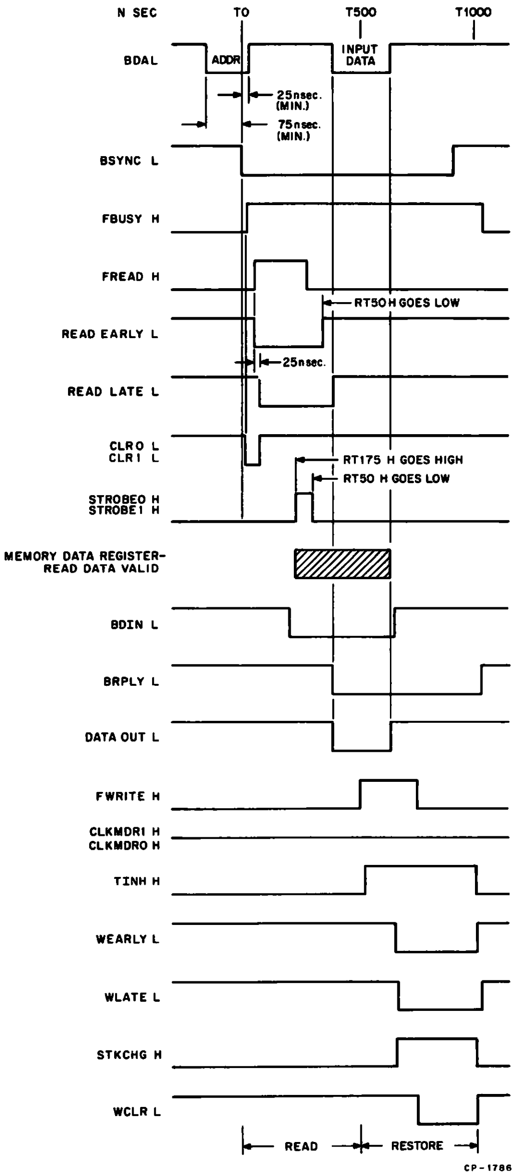


Figure MMV11-A-8
Read-Restore Memory Cycle Timing

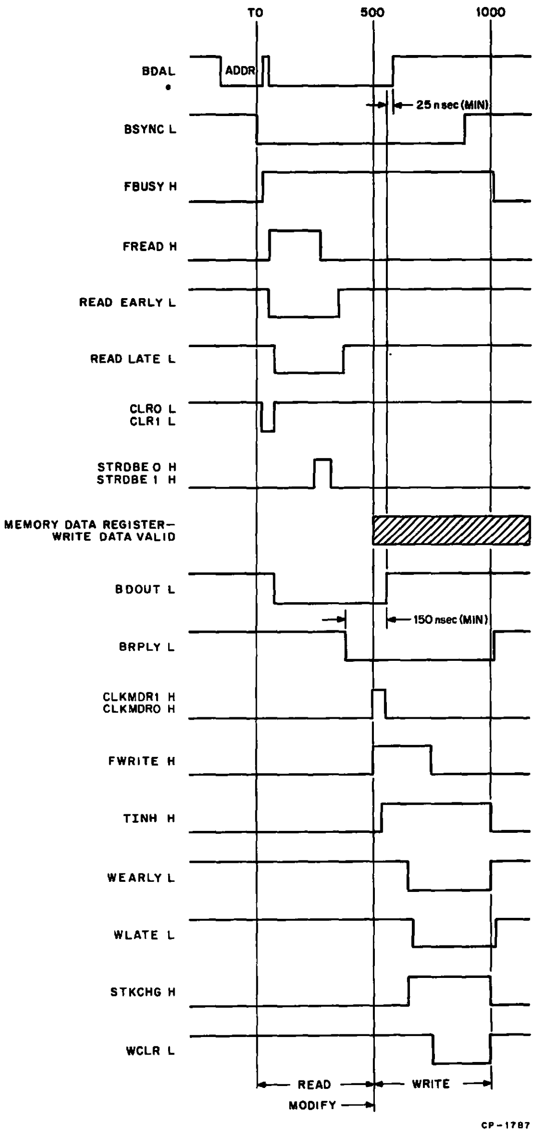


Figure MMV11-A-9
Read-Modify-Write Memory Cycle Timing

MMV11-A

The bus master device initiates the data transfer portion of the DATI transaction by asserting BDIN L. The reply enable flip-flop is set on the trailing edge of RT100 H 375 ns after BSYNC L. If RDIN H (BDIN L inverted) is received earlier than 375 ns after BSYNC L, the reply flip-flop input gates wait 375 ns to produce an active RPLY SET L signal (Figure 8), which direct-sets the reply flip-flop and produces the active FRPLY H and BRPLY L signals. FRPLY L is gated with RDIN L and inverted, producing the DATA OUT L signal which gates memory data register bits onto the BDAL bus. If RDIN H is received later than 375 ns after BSYNC L, the reply flip-flop sets on the leading edge of RDIN H. The trailing edge of RT150 H (T425 L) is gated with RPLY SET L, producing a write initiate pulse which clocks the high FBUSY H signal into the write flip-flop, initiating the restore portion of the memory cycle.

Restore timing is produced by the write time generator in a manner similar to that described for read time generation. At W700 H time, TINH H and TINH L (475 ns pulses) are produced for the inhibit drivers. TINH H also inhibits the reply clear gate, and the reply flip-flop remains set for the remainder of the memory cycle. WEARLY L and STK CHG H go active on the leading edge of WT175 H and remain active for 350 ns. Similarly, WLATE L goes active on the leading edge of WT175 H and remains active for 325 ns. At WT250 H time, WCLR L is produced, clearing the reply enable and erite flip-flops; thus, write time generator outputs are 250 ns pulses. Memory data is restored (written) during the time that TINH H, WEARLY L, and WLATE L are active. The memory cycle terminates when both SYNC L and FRPLY L go to their passive states. The busy clear gate detects this condition, producing a low pulse which clears the busy flip-flop, and the memory cycle ends.

The DATO cycle is similar to the DATI cycle except that during the addressing portion of the bus cycle, the bus master device asserts BWTBT L. RWBT H goes high, and the leading edge of SYNC H clocks the byte flip-flop to the set state. The active FWBT L signal is only used when in the write portion of the DATIO cycle, as described later. However, during a DATO bus transaction, RDIN H is not received; instead, RDOUT H is received, enabling the REPLY SET L gates, as shown in Figure 9. RDOUT enables one input to the WRITE TIME L gate. At the same time that the write flip-flop clocks to the set state, WRITE TIME L goes low, enabling CLK MDR0 and 1 H gates. Since a DATO bus cycle is in progress, BWTBT L remains passive during the data transfer portion of the bus cycle. Hence, RWBT H is low, WRITE WORD H is high, and the two byte select OR gates apply low signals to the remaining CLK MDR gates. CLK MDR 0 and 1 H then clock the BDAL bus data into the memory data register; the previously read data is lost. The write portion of the cycle continues as described for the restore portion of the DATI operation.

When executing a DATOB bus transaction, BWTBT L and RWBT H remain active for the duration of the bus cycle. Hence, the WRITE WORD H signal remains passive. The byte select flip-flop that stores byte addressing bit RA0 H during addressing time enables generation of only one CLK MDR H signal. When RA0 H is low, FA0 L goes high and CLK MDR 0 H clocks low byte data bits from only BDAL0–7 L into the memory data register. Register bits 8–15 remain unchanged. Similarly, when RA0 H is high, FA0 H goes high and CLK MDR 1 H clocks high byte data bits from only BDAL8–15 L into the memory data register. Register data bits 0–7 remain unchanged. The write portion of the memory cycle then continues as previously described.

When executing a DATIO bus cycle, two complete memory cycles are executed. They include a DATI and a DATO or DATOB cycle as previously described. However, when executing a DATIO bus transaction, BSYNC L remains active for the duration of the transaction. Hence, SYNC H, which generates FBUSY L during the read-restore portion of the cycle, cannot initiate the second read-modify-write memory cycle. Instead, FWBT L, stored during the addressing portion of the cycle, enables a read initiate pulse on the leading edge of RDOUT H. The read flip-flop goes to the set state and operation continues as described for DATO or DATOB bus transactions.

DC Protection and V_{CC} Switch

DC protection and V_{CC} switch circuits are shown in Figure 10. The dc protection circuit is activated during power-fail or bus initialize conditions. BDCOK H and BINIT L are inverted and ORed to produce LOCKOUT L. Normally, this signal is passive (high), enabling bank addressing and resulting in an active DSEL H signal when the memory is addressed. However, if BDCOK H goes low (power-fail) or BINIT L is asserted low, LOCKOUT L immediately inhibits the bank addressing function.

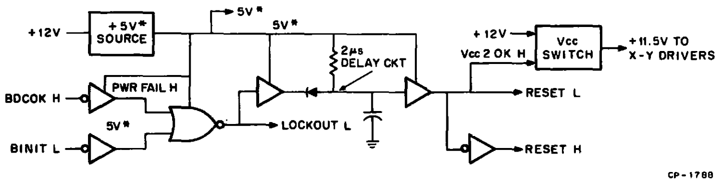


Figure MMV11-A-10
DC Protection and V_{CC} Switch Circuits

WARNING

The program must terminate and issue a HALT instruction within 2 ms of the time that the BPOK signal indicates that a power failure has occurred. Failure to do so could result in the loss of data in one or more memory locations.

The reset signals are also generated by this circuit. RESET L goes active (low) whenever LOCKOUT L is active. A 2 μs delay circuit enables the memory to complete its present cycle before RESET. RESET L is also inverted to produce RESET H; both signals are used to clear (initialize) memory timing control circuits.

To produce a 5 V* source for reset circuits and bus receivers BSYNC L, BDIN L, BDOUT L, BWTBT L, and BREF L, +12 V power is required. Thus, if +12 V is removed, all MMV11 memory operations are disabled. However, if +5 V is removed and the +12 V remains, the 5 V* allows memory protect logic to remain functional.

RESET L is also applied to the VCC2OK H input to the V_{CC} switch circuit. This signal is high only when both +5 V and +12 V power sources are normal. The V_{CC} switch comprises a transistor (V_{CC} switch), which is turned on when power is normal to produce +11.5 V power for X-Y driver circuits.

DC-DC Inverter

The dc-dc inverter circuit is shown in Figure 11. It is comprised of an inverter oscillator using a saturable transformer, a negative rectifier, and a filter. A 3-terminal regulator chip produces the regulated −5 V for sense amplifier operation.

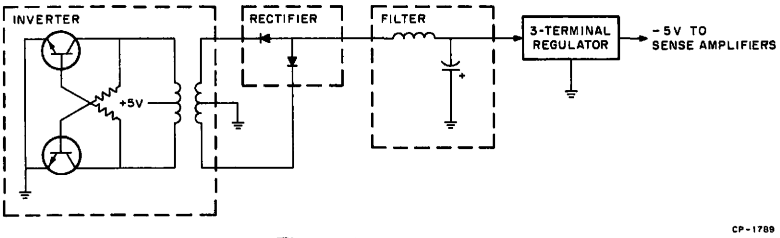


Figure MMV11-A-11
DC-DC Inverter Circuit

MRV11-AA 4K BY 16-BIT READ-ONLY MEMORY

GENERAL

The MRV11-AA is a basic read-only memory module on which the user can install programmable read-only memory (PROM) or masked read-only memory (ROM) chips.

FEATURES

- 4096 by 16-bit capacity using 512 by 4-bit chips, or 2048 by 16-bit capacity using 256 by 4-bit chips
- Compatibility with chips available from multiple sources
- Jumpers that allow the user to select the 4K memory address space to which the MRV11-AA will respond, chip type, and upper or lower 2K segment (when 256 by 4-bit chips are used)

SPECIFICATIONS

Identification	M7942
Size	Double
Power	
4K X 16 ROM less PROM integrated circuits	+5 V ± 5% at 0.4 A
Thirty-two 512 X 4 PROM integrated circuits	+5 V ± 5% at 2.8 A
Bus Loads	
AC	1.8
DC	1.0

CONFIGURATION

General

Depending on PROM type, the module's capacity is either 4096 16-bit words or 2048 16-bit words, using 512 by 4-bit or 256 by 4-bit PROMs, respectively. Full address decoding is provided on the module. The user can select the 4K address bank in which the module resides by installing (or removing) jumpers on the module. Similarly, when using 256 by 4-bit PROMs, the user can jumper-select the upper or lower 2K segment within the selected 4K address bank. Note that 512 by 4-bit and 256 by 4-bit PROMs cannot be mixed on a MRV11-AA module; the user configures jumpers on the module for the PROM type being used.

A partial listing of manufacturer's PROMs that will operate in the MRV11-AA is given in Table 1.

MRV11-AA

Table 1 MRV11-AA PROM Types

Manufacturer or Source	512 by 4-Bit PROMs	256 by 4-Bit PROMs
Digital Equipment Corp	MRV11-AC	–
Intersil	IM5624	IM5623
Signetics	82S131	82S129
MMI	6306	6301

PROMs used must be tri-state output devices that conform to the device pinning, data, and addressing described herein.

The user can install PROMs in increments of four each. When using 512 by 4-bit PROMs, memory expansion is in 512-word increments. When using 256 by 4-bit PROMs, memory expansion is in 256-word increments. Jumpers on the MRV11-AA can be cut by the user to prevent an incorrect BRPLY L signal from being generated when unpopulated locations are addressed on the module.

The following information will enable the user to prepare the MRV11-AA for use (jumper-selected addressing and PROM type selection) and includes information required for correct PROM and ROM programming.

PROM Type Jumpers

The module is supplied with jumpers W8, W9, and W10 installed for use with 512 by 4-bit PROMs. When using 256 by 4-bit PROMs, W8, W9, and W10 must be cut or removed and jumpers W11 and W12 installed; in addition, either W13 (lower 2K) or W14 (upper 2K) must be installed to properly address the lower 2K or upper 2K address segment within the 4K memory bank. Jumpers are located as shown in Figure 1.

Address and Reply Jumpers

The user must consider both 4-bank address selection and BRPLY L signal generation when configuring a module for use. PROMs are arranged in eight physical rows (CE0–CE7) of four each. Entire rows can be unpopulated, allowing those addressed locations to be used by read/write memory contained on another module. When this is done, the BRPLY L jumpers (W0–W7) associated with the unused rows should be cut or removed to prevent the MRV11-AA from returning a BRPLY L signal when those rows are addressed. A listing of octal addresses (within a 4K bank), physical rows, and BRPLY L jumpers is provided in Table 2; use data listed for the PROM type being used.

MRV11-AA

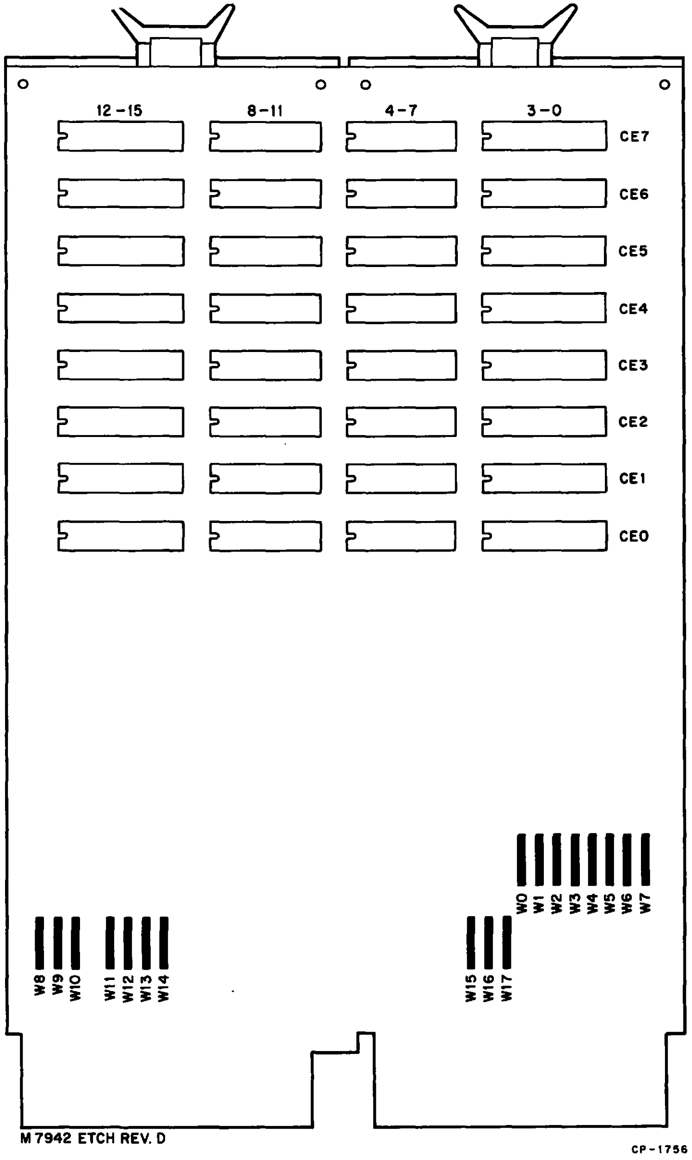


Figure MRV11-AA-1
MRV11-AA Jumper Locations

Table 2 PROM/ROM Addressing Data

4K Bank Selection

W15*	W16*	W17*	Bank	Word/Byte Address Range
I	I	I	0	0–17777
I	I	R	1	20000–37777
I	R	I	2	40000–57777
I	R	R	3	60000–77777
R	I	I	4	100000–117777
R	I	R	5	120000–137777
R	R	I	6	140000–157777
R	R	R	7	160000–177777

*R = jumper removed, I = jumper installed

512 by 4-Bit PROM Addressing Within a Bank

NOTE

Jumpers W8, W9, W10 are installed; W11, W12, W13, W14 are removed.

Reply Jumper*	Physical Row	Prom Octal Address Range
W0	CE0	0–1777
W1	CE1	2000–3777
W2	CE2	4000–5777
W3	CE3	6000–7777
W4	CE4	10000–11777
W5	CE5	12000–13777
W6	CE6	14000–15777
W7	CE7	16000–17777

*Jumper installed = BRPLY L enabled; jumper removed = BRPLY L not enabled.

MRV11-AA

Table 2 PROM/ROM Addressing Data (Cont)

256 by 4-Bit PROM Addressing Within Lower 2K Portion of Bank

NOTE

Jumpers W11, W12, W13 are installed; W8, W9, W10, W14 are removed.

Reply Jumper	Physical Row	PROM Octal Address Range
W0	CE0	0–777
W4	CE4	1000–1777
W1	CE1	2000–2777
W5	CE5	3000–3777
W2	CE2	4000–4777
W6	CE6	5000–5777
W3	CE3	6000–6777
W7	CE7	7000–7777

256 by 4-Bit PROM Addressing Within Upper 2K Portion of Bank

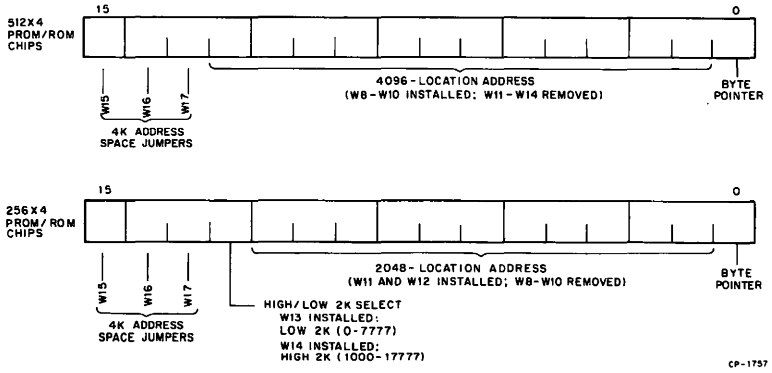
NOTE

Jumpers W11, W12, W14 are installed; W8, W9, W10, W13 are removed.

Reply Jumper	Physical Row	PROM Octal Address Range
W0	CE0	10000–10777
W4	CE4	11000–11777
W1	CE1	12000–12777
W5	CE5	13000–13777
W2	CE2	14000–14777
W6	CE6	15000–15777
W3	CE3	16000–16777
W7	CE7	17000–17777

MRV11-AA

The 4K bank in which the MRV11-AA resides is programmed by connecting bank address jumpers W15–W17, as appropriate. The module is supplied with all bank address jumpers installed (bank 0). Jumpers installed represent logical 0s; jumpers not installed represent logical 1s. Figure 2 illustrates addressing words used with the MRV11-AA. Refer to the addressing format for the type of PROMs or ROMs being used.



512X4 PROM/ROM CHIPS
15
0
4096-LOCATION ADDRESS (W8-W10 INSTALLED; W11-W14 REMOVED)
BYTE POINTER
W15
W16
W17
4K ADDRESS SPACE JUMPERS
256X4 PROM/ROM CHIPS
2048-LOCATION ADDRESS (W11 AND W12 INSTALLED; W8-W10 REMOVED)
HIGH/LOW 2K SELECT
W13 INSTALLED: LOW 2K (0-7777)
W14 INSTALLED: HIGH 2K (1000-17777)
CP-1757

Figure MRV11-AA-2
MRV11-AA Address Word Formats

PROM Integrated Circuits

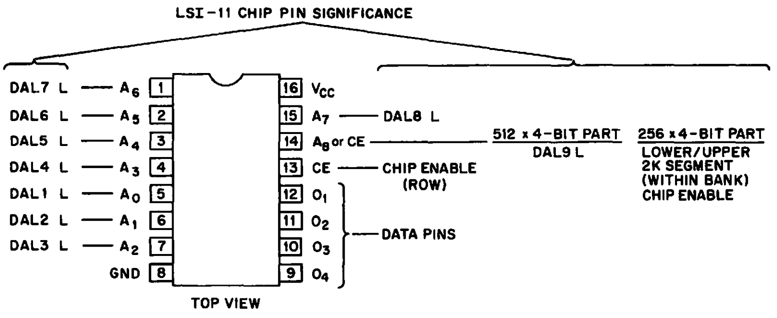
The actual procedure for loading data into PROMs (or writing specifications for masked ROMs) will vary, depending on the manufacturer. Those procedures are beyond the scope of this document. (See PROM/ROM manufacturer's data sheets.) However, the user must be aware of the PROM pins versus LSI-11 data bit relationship, and the pins versus memory address bits. Address and data pins are described below.

As previously discussed, PROMs are arranged in rows of four each. Each PROM contains locations of four bits. Hence, four PROMs are used to provide the 16-bit data word formats for each row. Rows are designated by their respective chip enable (CE0–CE7) signals. Depending on the PROM type used, a row of four PROMs contains 512 or 256 16-bit read-only memory locations. The actual PROM within a row is designated by one additional digit (0, 1, 2, or 3). Hence, the data pins are assigned to LSI-11 bus bits as listed in Table 3.

Table 3 Data Pin Assignments

PROM Pin	PROM 0	PROM 1	PROM 2	PROM 3
9	BDAL3	BDAL7	BDAL11	BDAL15
10	BDAL2	BDAL6	BDAL10	BDAL14
11	BDAL1	BDAL5	BDAL9	BDAL13
12	BDAL0	BDAL4	BDAL8	BDAL12

Addressing of PROMs is shown in Figure 3. All PROMs used on the MRV11-AA must conform to this information. Observe that the only difference between 512 by 4-bit and 256 by 4-bit PROM pins is pin 14. The 512 by 4-bit part uses this pin for address bit DAL9; the 256 by 4-bit part uses this pin for a chip enable when both bank address and 2K segment address are true. Also note that bus address bits do not follow in sequence with PROM manufacturer's address designations. The pinning arrangement shown allows for the use of commonly available PROMs and ROMs and optimum (compact) MRV11-AA module layout.



NOTE:
Designations immediately adjacent to pins are typical designations used by chip manufacturers — not LSI-11 designations. LSI-11 designations for correct addressing are located away from the chip. Observe that these signals are low — active; they are double-inverted bus signals (low = logical "1").

IC-0169

Figure MRV11-AA-3
PROM/ROM Pin Addressing

Programming PROMs
Complete information for programming PROMs is contained in Chapter 3. Do not attempt to program PROMs until you are thoroughly familiar with the information contained in that chapter.

FUNCTIONAL DESCRIPTION

General
Major functions contained on the MRV11-AA module are shown in Figure 4. ROM data stored on the module can be addressed and read by the processor or other DMA devices by executing a DATI bus cycle. Data/address lines BDAL0–15 L and three bus interface control signals (BSYNC L, BDIN L, and BRPLY L) comprise all interface signals required for accessing the read-only memory. BREF L inhibits BRPLY L and BDAL bus drivers during memory refresh operations.

Addressing
A master device can address any 16-bit word in the 4K module by placing appropriate address bits on BDAL1–15 L during the addressing portion of the DATI cycle. BDAL0 is not used on the MRV11-AA since this address bit functions only as a byte pointer during DATOB and the write portion of DATIOB bus cycles. Bus receivers route DAL13–15 H to the bank select decoder and DAL1–12 H to the address storage latch. Bank selection occurs when the 4K address encoded on DAL13–15 H is equal to the user-configured value selected by jumpers W17–W15. The resulting bank select (BS H) and address bits DAL13–15 H are then stored in the address storage latch on the leading edge of BSYNC L. Stored address bits SA1–8 H are buffered to produce BA1–9 L, which are applied to all ROM/PROM chips on the module.

When 512 by 4-bit chips are used, SA9 H is routed via jumper W10 to a buffer, producing the inverted BA9 L address bit for all chips (pin 14). However, when 256 by 4-bit chips are used, W10 is removed and W12 is connected, forcing a low (chip enable) signal to be applied to all chips (pin 14); note that 256 by 4-bit chips do not receive address bit 9.

Memory chip sockets are arranged in eight physical rows of four sockets each. The memory is expanded by installing all four chips in each desired row. Four chips provide the full 16-bit word storage for LSI-11 instructions and data. Only one row is enabled by a chip enable (CE) signal, produced by chip row select logic and chip type jumpers.

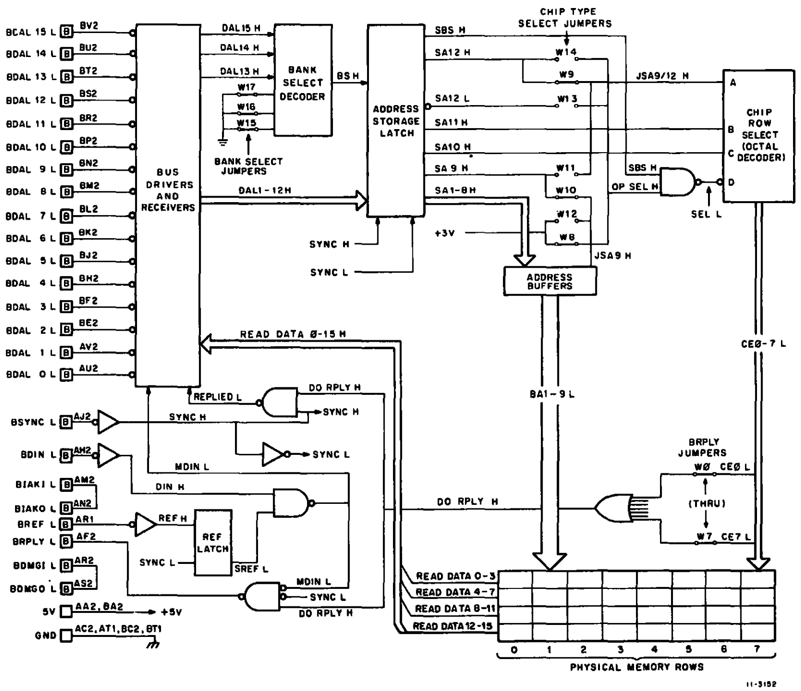


Figure MRV11-AA-4
MRV11-AA Logic Block Diagram

When 512 by 4-bit chips are used, jumpers W8, W9, and W10 are installed. The chip row select octal decoder receives stored address bits SA10, SA11, and SA12 on its A, B, and C inputs, respectively, as shown in Figure 5. Bank select stored (SBS H) is gated to produce a low SEL L enable signal, which is applied to the D input of the decoder. (The decoder is actually a decimal decoder; whenever a high signal is applied to its D input, outputs 0–7 are inhibited.) One decoder output goes low, enabling the appropriate physical row addressed by bits SA10–12 L.

When 256 by 4-bit chips are used, jumpers W8, W9, and W10 are removed and jumpers W11, W12, and either W13 or W14 are installed, as shown in Figure 6. SA10 and SA11 are applied to octal decoder A and B inputs, respectively. Bit SA9, which is not used to directly address the 256 by 4-bit chips, is then applied to input C of the octal decoder.

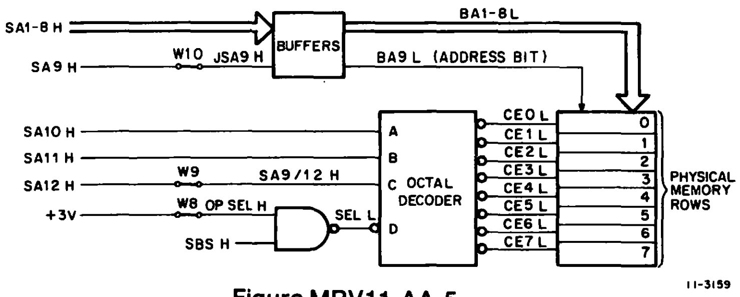


Figure MRV11-AA-5
Figure 5 512 by 4-Bit Chip-Jumper Configuration

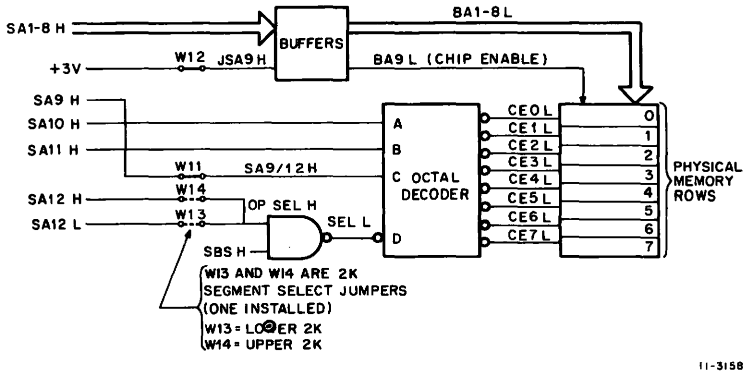


Figure MRV11-AA-6
256 by 4-Bit Chip-Jumper Configuration

SA12 H and SA12 L are available for jumper selection of the desired 2K segment within the 4K bank. W13, when installed, selects the lower 2K; W14 selects the upper 2K. When the selected segment is addressed, OP SEL goes high. This signal is gated with SBS H to produce the low (active) octal decoder enable signal.

Caution must be used when assigning memory to bank 7 to avoid conflicts with preassigned device addresses. This 28–32K address space is normally used for peripheral device addresses. Certain DIGITAL-supplied system programs and operating systems determine the presence or absence of some of these devices by accessing the assigned locations; if a response is obtained (i.e., no bus time-out occurs), the program assumes that the device is present. Thus, having a memory respond to any of these preassigned locations will give the erroneous indication that the corresponding device is installed in the system.

Data Read Operation

Once the ROM/PROM chip sockets are addressed, the data can be read by the bus master device. Data is available within 120 ns after BSYNC L is received. One active CE0–7 L signal produces the active DO RPLY H signal, which enables reply and DBAL bus driver gating. Active DO RPLY H and SYNC H signals are gated, producing the REPLIED L signal, which enables one of the two bus driver enable inputs. The remaining enable input is MDIN L. The bus master device asserts BDIN L to request the data. DIN H is ANDed with the passive (high) SREF L signal, producing MDIN L, and read data is enabled onto BDAL0–15 L. Active MDIN L, SYNC L, and DO RPLY H signals also enable the BRPLY L bus driver, producing the required response to BDIN L.

When the system is in a memory refresh operation, the MRV11-A must not respond to the BSYNC/BDIN refresh bus transactions. BREF L is asserted during the addressing portion of the bus cycle and the refresh latch stores REF H on the leading edge of SYNC L. SREF L goes low and inhibits the MDIN L signal. Hence, BDAL and BRPLY L bus drivers are not enabled.

I/O Timing and Bus Restrictions

Addressed memory read data is available within 120 ns after the BSYNC L signal is received by the MRV11-AA. Logic on the module responds to DATI bus cycles only. DATO or DATOB bus cycles will result in a bus time-out error. Logic functions on the module are not affected by the bus initialize (BINIT L) signal.

MRV11-BA LSI-11 UV PROM/RAM

GENERAL

The MRV11-BA is a memory option that contains eight sockets in which MRV11-BC ultraviolet (UV), erasable, programmable read-only memory (PROM) integrated circuits can be installed.

The MRV11-BA also contains 256 by 16-bit static random access memory (RAM) that can be used as a "scratchpad" and "stack" by system software. The RAM contents are volatile; that is, when operating power is removed, memory data is lost. PROM contents are not volatile; programs and data stored in PROMs are available when operating power is restored.

Each MRV11-BC PROM option includes one 1024 (1K) by 8-bit unprogrammed UV PROM integrated circuit (Intel 2708 PROM). UV PROMs can be erased by exposure to high-intensity ultraviolet light and then reprogrammed with new programs and data. A clear quartz window over the PROM chip allows the ultraviolet light to be directed onto the chip. Optional QJV11 ROM/PROM formatter software is available for conversion of absolute loader format programs into listings and paper tapes in PROM content format.

FEATURES

- On-board 256-word static RAM
- Sockets provided for installation of up to 4K words (8K bytes) of PROM in 1K increments
- PROM and RAM address space can be independently customer configured via jumpers.
- No special power is required. Only the normal +5 and +12 Vdc operating voltages present on the LSI-11 bus are required. An onboard "charge pump" circuit provides the necessary –5 V operating voltage to the PROM array.
- Completely compatible with LSI-11 bus protocol

MRV11-BA

SPECIFICATIONS

Identification	M8021
Size	Double
Power	
LSI-11 UV PROM less PROM integrated circuits	+5 V ± 5% at 0.58 A +12 V ± 3% at 0.34 A
With eight 1K X 8 PROM integrated circuits	+5 V ± 5% at 0.62 A +12 V ± 3% at 0.5 A
Bus Loads	
AC	2.8
DC	1.0

CONFIGURATION

General

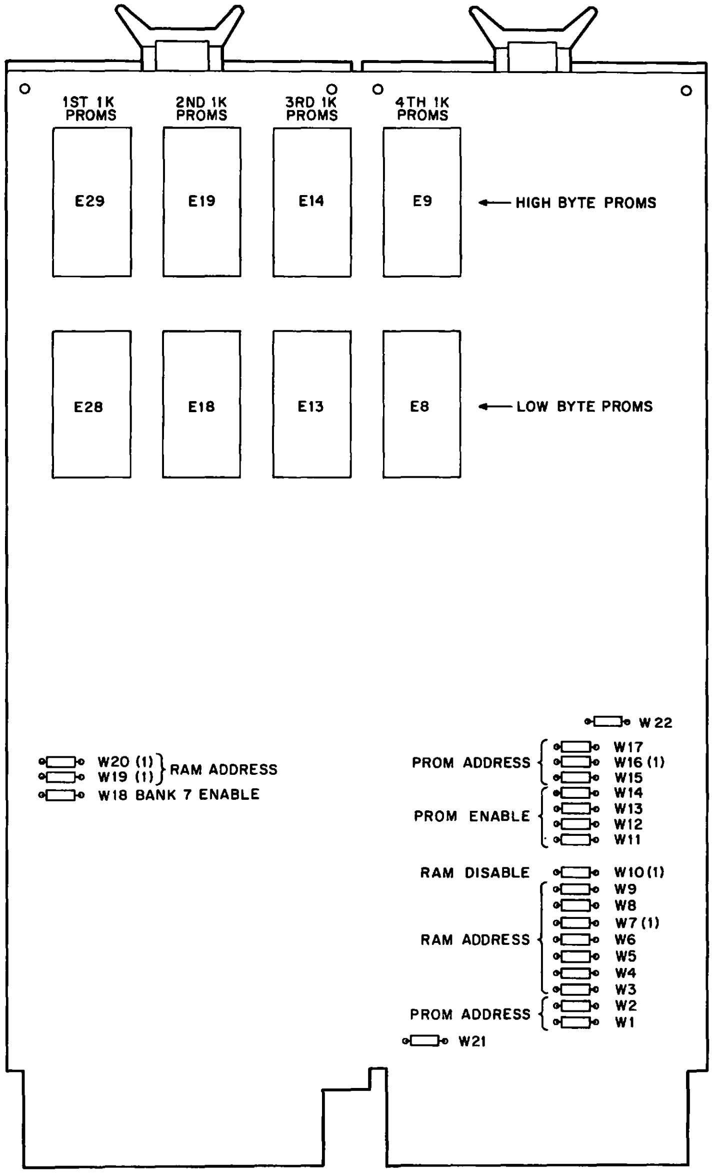
Jumper locations are included on the MRV11-BA module as shown in Figure 1. Jumpers allow independent selection of system memory starting addresses for the RAM and PROM memory functions. In addition, four special jumper locations (W10, W18, W21, and W22) are provided.

W10, when installed, disables the 256 RAM portion of the MRV11-BA when the RAM function is not desired. W18, when installed, enables PROM and/or RAM operation in bank 7 (the 4K memory addresses ranging from 160000 through 177777). Bank 7, by PDP-11 convention, is normally reserved for peripheral devices, and system memory would normally be configured for addresses ranging from 0 through 157777. The MRV11-BA is factory configured with W10 removed and W18 installed.

W21 and W22 control the MRV11-BA response to attempts to "write" in PROM locations. The module is factory-configured with W21 installed and W22 removed. When configured in this manner, any attempt to write in the PROM will result in a bus time-out error.

W21 can be removed and W22 can be installed to enable "pseudo-write" operations in PROM locations. Note that this jumper configuration only prevents bus time-out errors; it is not possible to actually write into (output data to) PROM locations. This jumper configuration is required to support the following instructions.

Mnemonic	Octal Code	Instruction
MTPS	1064SS	Move byte to PS
MUL	070RSS	Multiply
DIV	071RSS	Divide
ASH	072RSS	Shift arithmetically
ASHC	073RSS	Arithmetic shift combined



NOTE:
(1) = JUMPERS NOT FACTORY INSTALLED.

MR-0104

Figure MRV11-BA-1
MRV11-BA Jumper and Socket Locations

All of the instructions listed require DATIO (read-modify-write) bus cycles. If the source operand (SS) refers to a PROM location, the MRV11-BA must be configured with W21 removed and W22 installed in order to avoid bus time-out errors. See Chapter 3 for additional details.

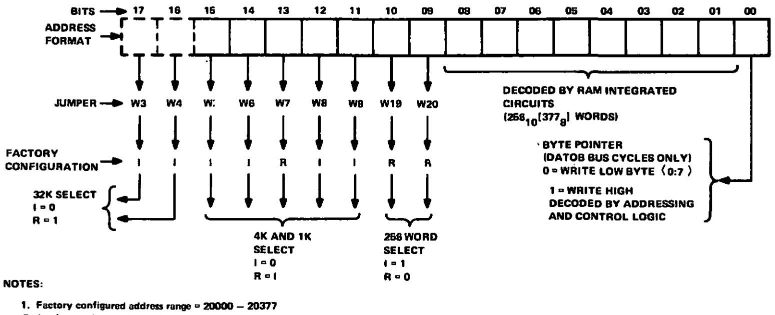
Address selection jumpers allow PROM and RAM addressing through a 128K address range. Bank 7 is the highest 4K portion of the address range. RAM addresses can reside within a populated PROM bank address. When this is done, RAM data will be properly accessed and PROM contents are not enabled. Detailed instructions for configuring address jumpers are provided below.

NOTE

System memory must include memory location 000004. This location may be either read-only or read-write memory. The processor executes a dummy read bus cycle during the power-up sequence using this address and requires a reply to complete the bus cycle. The actual memory contents read from the location are not used and can be any value.

RAM Address Jumpers

RAM addresses can be located in any 256-word portion of system memory, starting at 256-word-segment boundaries. The relationship between bus address bits and jumpers 19, 20, and 3 through 9 is shown in Figure 2. Configure the RAM starting address by removing and/or installing the appropriate jumpers.



BITS
17
16
15
14
13
12
11
10
09
08
07
06
05
04
03
02
01
00
ADDRESS FORMAT
JUMPER
W3
W4
W:
W6
W7
W8
W8
W19
W20
DECODED BY RAM INTEGRATED CIRCUITS (256_{10}[377_8] WORDS)
FACTORY CONFIGURATION
I
I
I
I
R
I
I
R
R
BYTE POINTER (DATOB BUS CYCLES ONLY) 0 = WRITE LOW BYTE (0:7) 1 = WRITE HIGH DECODED BY ADDRESSING AND CONTROL LOGIC
32K SELECT I = 0 R = 1
4K AND 1K SELECT I = 0 R = I
256 WORD SELECT I = 1 R = 0

NOTES:

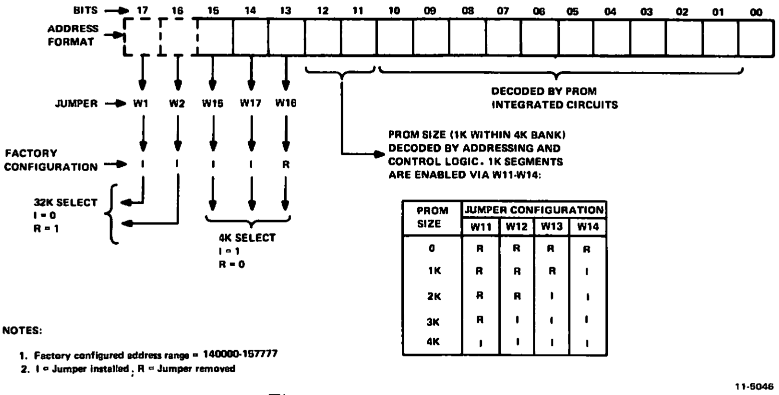
1. Factory configured address range = 20000 – 20377
2. I = Jumper installed; R = Jumper removed
3. W10 removed = RAM ENABLE
   W10 installed = RAM DISABLE

11-5045

Figure MRV11-BA-2
MRV11-BA RAM Addressing

PROM Address Jumpers

PROM addresses can be located in any 4K bank of system memory. The relationship between bus address bits, PROM size, and jumpers is shown in Figure 3. Configure the PROM starting address by removing and/or installing the appropriate jumpers. Remove or install PROM size jumpers W11 through W14 as shown in the figure; these jumpers must be removed to conform to PROM size (in increments of 1K) to prevent erroneous addressing of unpopulated sockets. The MRV11-BA is factory-configured with W11 through W14 installed.



PROM SIZE	JUMPER CONFIGURATION			
	W11	W12	W13	W14
0	R	R	R	R
1K	R	R	R	I
2K	R	R	I	I
3K	R	I	I	I
4K	I	I	I	I

NOTES:

1. Factory configured address range = 140000-157777
2. I = Jumper installed; R = Jumper removed

Figure MRV11-BA-3
MRV11-BA PROM Addressing

MRV11-BC Handling Precautions

MRV11-BC integrated circuit PROMs are metal oxide semiconductor (MOS) devices that can be damaged through improper handling. MOS devices can be easily damaged by static discharges due to their high input/output impedance. Safe installation requires that the conductive foam in which the chip is shipped be brought into physical and electrical contact with the MRV11-BA module or PROM programming equipment prior to removing the PROM from the foam. Unnecessary handling of PROMs should be avoided once removed from the foam. When programmed and installed in MRV11-BA sockets, there is no danger of static discharge damaging the PROMs.

Each MRV11-BC PROM is implemented in a 24-pin integrated circuit package. Mechanical damage to the PROMs can occur if they are carelessly handled. When installing PROMs, ensure that all pins are properly started into the socket before pressing the PROM pins all the way into the socket.

MRV11-BA

An instruction sheet illustrating proper handling procedures is included with each purchase of MRV11-BC PROMs. Refer to that sheet for PROM installation and removal instructions.

Installing the MRV11-BA Module
The MRV11-BA module can be installed in any LSI-11 bus. It only requires one option location and is not dependent on position (device priority) along the bus. Hence, the module can be installed in any option location in single and multiple backplane systems. The module requires no special power; all operating power (+5 V and +12 V) is supplied by the normal power present on the backplane. The MRV11-BA normally should not be configured for "pseudo-write" operation if it is being used in a DIGITAL operating system. The software may attempt to write in PROM locations resulting in bus time-out errors.

FUNCTIONAL DESCRIPTION

General
Four major functions comprise the MRV11-BA option: addressing and control logic, 4K X 16 PROM array, 256 X 16 RAM array, and charge pump circuit. These functions are shown in Figure 4. The PROM and RAM arrays comprise the actual memory portion of the module. The MRV11-BA option contains factory-installed RAM integrated circuits; PROM integrated circuits (MRV11-BC) are optional, and must be programmed prior to installation on the module. The charge pump circuit is a dc-dc voltage converter that produces –5 V operating power for the PROM array. Each function is described in the following paragraphs.

PROM Array
Optional PROMs comprise the PROM array shown in Figure 5. Each MRV11-BC PROM includes one 1K X 8 PROM integrated circuit. When two options are installed, they comprise a 1K X 16 read-only memory. The MRV11-BA option is expanded to 4K PROM by installing eight MRV11-BC options.

Four chip enable signals [CE (0:3) L] select the addressed pair of 1024-location by 8-bit (1K X 8) PROMs. Only one chip enable signal will go active when the PROM array is addressed, selecting a 1K portion of the 4K array. Addressing within the selected 1K portion is controlled by buffered address signals BA (1:10) H. Addressing and control logic functions control the chip enable and buffered address signals, and place the PROM output data DAL (0:15) on the LSI-11 bus where it can be read by the bus master. When not addressed by a chip enable signal, the PROM chip outputs go to a high-impedance state, effectively disconnecting the PROM array from the DAL signal lines.

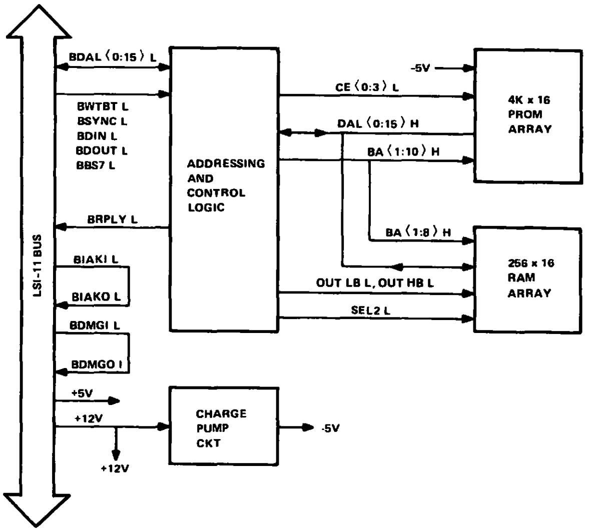


Figure MRV11-BA-4
MRV11-BA Block Diagram

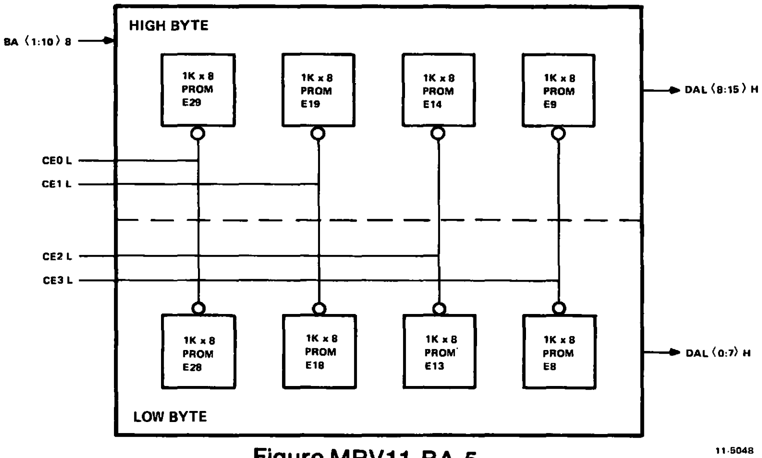


Figure MRV11-BA-5
PROM Array

MRV11-BA

RAM Array

Four factory-installed 256-location by 4-bit (256 X 4) RAM integrated circuits comprise the RAM array, as shown in Figure 6. SEL2 L is asserted low by the addressing and control logic whenever the RAM array is addressed. When the RAM array is not addressed, SEL2 L goes high and the RAM input/output data pins [DAL (0:15) H] go to a high-impedance state, effectively disconnecting the array from the DAL lines.

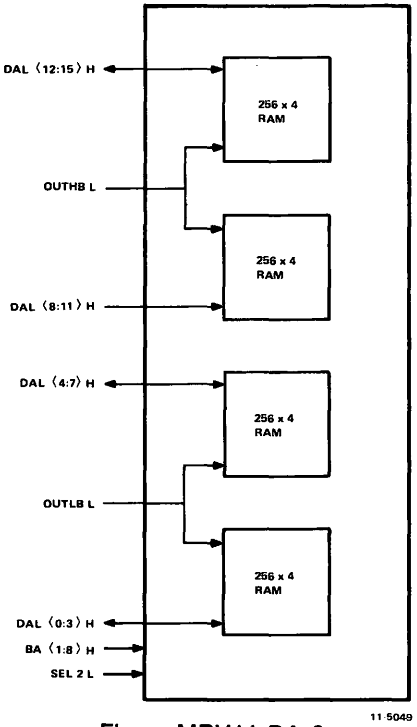


Figure MRV11-BA-6
RAM Array

When addressed, OUT HB L and OUT LB L select a read or write operation. When a read operation (DATI) is in progress, both OUT signals are high (write inhibit). During a 16-bit (word) write (DATO) operation, both signals are low. During an 8-bit (byte) write (DATOB) operation, only one OUT signal will go low, selecting the addressed byte.

Addressing and Control Logic

Addressing and control logic functions are shown in Figure 7. Separate address decoding logic is included for PROM and RAM arrays. A common PROM/RAM address latch stores buffered address bits BA (1:12) H for both memory functions. Protocol logic contained in one integrated circuit (type DC004) controls the MRV11-BA interface according to a strict LSI-11 bus protocol.

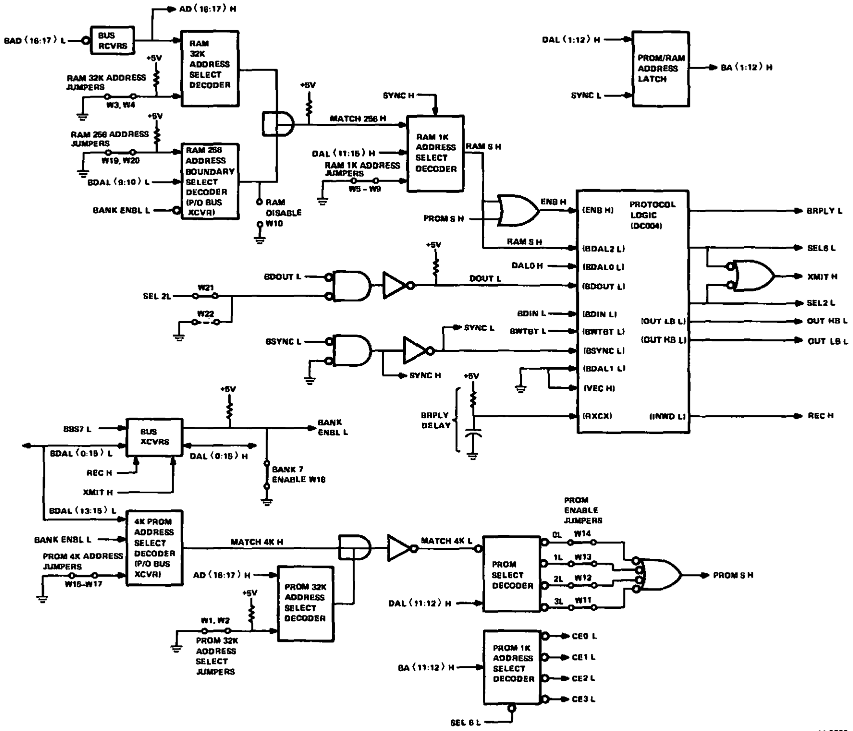


Figure MRV11-BA-7
MRV11-BA Addressing and Control Logic

The addressing and control logic also includes bus transceivers that receive and transmit address and data bits to and from the LSI-11 bus.

MRV11-BA

REC H, when high, inverts and gates BDAL (0:15) L bits onto the DAL (0:15) H lines. These lines comprise an internal 16-bit bidirectional data/address bus for the MRV11-BA module. When XMIT H is high and REC H is low, inverted DA (0:15) H bits are placed on BDAL (0:15) L.

The PROM address can be configured via jumpers W15–W17 to reside in any 4K bank of system memory. PROM 32K address select decoder and jumpers W1 and W2 permit addressing in 128K memory systems (presently not implemented in LSI-11 systems).

When a bus master device places a PROM address on the LSI-11 bus, MATCH 4K H goes high; this signal is inverted and applied to the PROM select decoder, enabling further address selection. The state of DAL (11:12) H determine which PROM select decoder output will go active (low). Jumpers W11 through 14 apply the active signal to the PROM S H OR gate. Only one signal will go active during a PROM read sequence, indicating the addressed 1K segment within the 4K bank. The jumpers can be removed to disable PROM S H when PROM sockets do not contain PROMs. PROM S H is ORed with RAM S H, producing ENB H. When active, ENB H indicates a valid address is present, enabling protocol logic operation. During the addressing portion of the bus cycle, BSYNC L goes active, latching the buffered address bits BA (1:12) H. BA (11:12) H are applied to the PROM 1K address select decoder, producing one active chip enable signal (CE0 L through CE3 L) that enables the appropriate pair of 1K X 8 PROM integrated circuits for the duration of the PROM read sequence. BA (1:10) H select the addressed location within the selected pair of 1K PROMs.

RAM addressing is accomplished by first decoding the active 32K portion of memory configured via W3 and W4, and the 256-word portion within a 1K segment configured via W19 and W20. When a bus master places an address on the bus that is within the configured 32K and 256-word address space, MATCH 256 H goes active (high), enabling the RAM 1K address select decoder. On the leading edge of SYNC H, the RAM 1K address select decoder latches the "match" states of MATCH 256 H, the address space configured via jumpers W5–W9, and 1K address bits on DAL (11:15) H. If the address is within the configured range, RAM S H goes active (high), producing active ENB H and RAM S H (BDAL 2 L) input signals for protocol logic operations. Word addressing within the 256-word address space is controlled by buffered address bits BA (1:8) H. In addition, during a write byte operation (DATOB), the protocol logic produces one active (low) OUT HB L or OUT LB L signal that selects the appropriate RAM integrated circuit to write (store) data; during a word write operation (DATO), both signals go active, enabling both RAM integrated circuits to write data. If RAM operation is not desired, RAM disable jumper W10 can be installed. When installed, W10 prevents MATCH 256 H from going active and RAM addressing cannot occur.

Bank 7 addressing is normally reserved for devices other than system memory. By PDP-11 convention, the upper 4K address space contains peripheral device addresses that are compatible with system hardware and software options. W18 is factory-installed and BANK ENBL L remains active, enabling all bank addresses, including bank 7. With bank 7 enable jumper W18 removed, an active BBS7 L (bank 7) bus signal causes BANK ENBL L to go high; at all other times (bank addresses other than bank 7), this signal remains low, enabling RAM and PROM address decoders.

PROM Read Sequence – The PROM read sequence is initiated when the LSI-11 bus master device places a valid address on the BDAL (0:15) H lines (Figure 8). A bank address falling within the user-configured 4K address space enables an active (high) MATCH 4K H signal. Similarly, the PROM 32K address select decoder enables MATCH 4K H when the LSI-11 bus address is within the configured 32K space. When both conditions are true, MATCH 4K H goes high. This signal is inverted, producing MATCH 4K L, enabling the PROM select decoder. The decoder decodes DAL (11:12) H address bits and produces one active (low) output that represents a 1K segment of the addressed 4K bank. The active signal is routed via an appropriate jumper (W11 through W14) to the PROM S H OR gate. Thus, the resulting active PROM S H signal signifies that a populated portion of PROM is being addressed.

The active PROM S H is ORed with the passive RAM S H signal, producing an active ENB H signal input to the protocol logic. The leading edge of BSYNC L then stores buffered address bits BA (1:12) H and causes protocol logic generation of an active (low) SEL6 L signal. SEL6 L produces an active XMIT H signal that enables the bus drivers in the bus transceivers; however, data is not actually placed on the bus until REC H goes low. SEL6 L also enables the PROM 1K address select decoder. Only one decoder chip enable output (CE0 through CE3) goes low, enabling the addressed pair of 1K by 8 PROMs to place read data on DAL (0:15) H lines. The active chip enable signal and buffered address bits BA (1:10) H thus complete the addressing portion of the PROM read sequence.

The bus master then asserts BDIN L to initiate the data portion of the sequence. The protocol logic responds by negating REC H and PROM data is placed on BDAL (0:15) L where it can be read by the bus master device. After a 600 ns delay from the leading edge of BDIN L, the protocol logic produces an active BRPLY L signal, indicating that the MRV11-BA has placed valid data on the bus. The bus master then reads the data and negates BDIN L. The protocol logic responds by terminating BRPLY L. Finally, the bus master terminates the bus cycle by negating BSYNC L. The protocol logic responds by producing an active (high) REC H signal, inhibiting bus transmitter portions and enabling bus

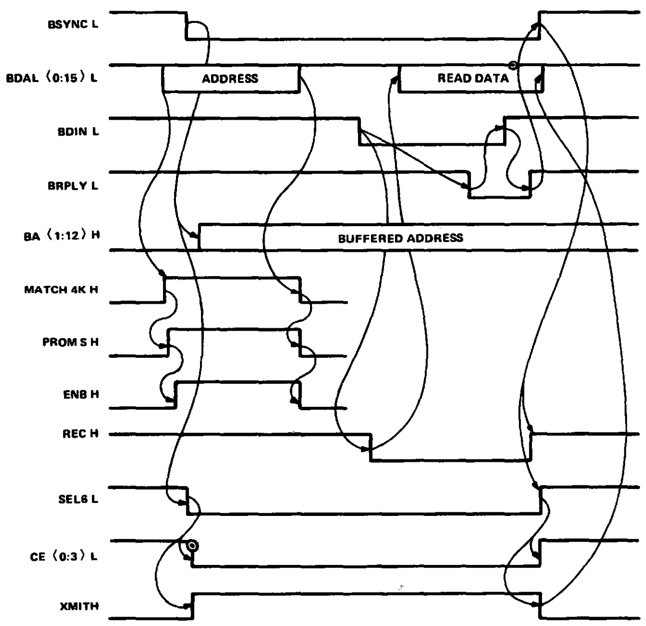


Figure MRV11-BA-8
PROM Read Sequence (DATI)

receiver portions of the bus transceivers, and negating SEL6 L. The passive SEL6 L signal inhibits PROM chip enable signal decoding and produces a passive XMIT H signal, and the PROM read sequence is completed.

PROM Reply to DATIO(B) Bus Cycles – The MRV11-BA module is factory-configured to reply only to DATI (read) cycles when PROM is addressed. However, in certain applications the reply to the PROM pseudo-write sequence may be required to prevent bus time-out errors. The module is factory-configured with W21 installed and W22 removed. This enables the DOUT L signal input to the protocol logic (Figure 8) only when the 256 RAM is addressed (SEL2 L is asserted low). When PROM is addressed, SEL2 L goes high and inhibits DOUT. Thus, attempting to write in PROM will result in bus time-out since DOUT is not received by the protocol logic.

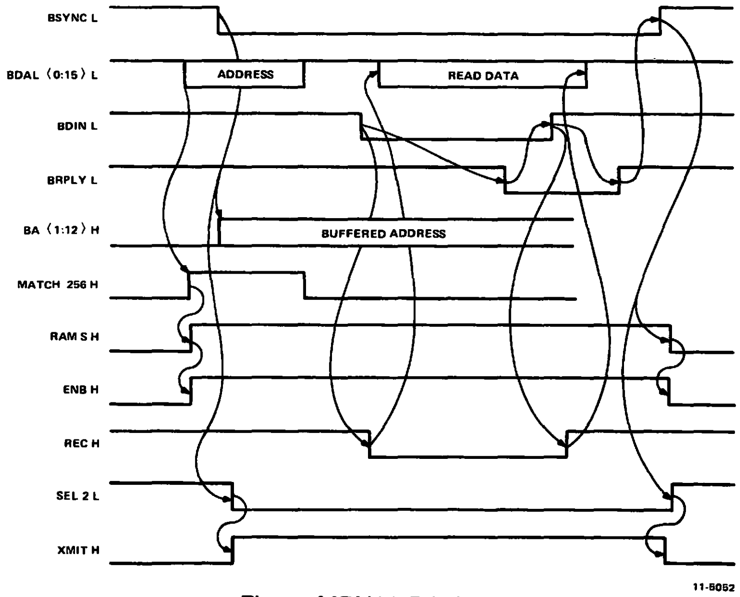


Figure MRV11-BA-9
RAM Read Sequence (DATI)

When reply to DOUT is required, W21 is removed and W22 is installed. Thus, the protocol logic receives DOUT during PROM pseudo-write sequences. Note that no useful function is performed by the protocol logic other than asserting BRPLY L to complete the bus cycle; thus, bus time-out errors are prevented.

RAM Read Sequence – A RAM read sequence is initiated when a bus master device places an address on the LSI-11 bus (Figure 9). The RAM 32K and 256 (word) address select decoders produce a high (active) MATCH 256 H signal if the address is within the user-configured 32K and 256 address space. MATCH 256 H enables the RAM 1K address select decoder. If the bus address bits [BDAL (11:15) L] are equal to the user-configured 1K address segment, RAM S H goes high (active), producing an active ENB H signal that enables protocol logic operation. The bus master then asserts BSYNC L, latching the state of RAM S H and buffered address bits BA (1:12) H; the protocol logic responds to BSYNC L by producing an active SEL2 L signal, and the addressing portion of the sequence is completed.

MRV11-BA

The active SEL2 L signal is applied to RAM integrated circuit chip enable inputs, enabling data to be read. Buffered address bits BA (1:8) H select the addressed word within the 256-word memory array. SEL2 L also produces an active XMIT H signal, enabling the transmit function in the bus transceivers; however, data is not placed on the BDAL (0:15) L bus until REC H goes low.

The bus master enters the data portion of the bus cycle by asserting BDIN L. MRV11-BA protocol logic responds to BDIN L by negating REC H and asserting BRPLY L 600 ns after the leading edge of BDIN L, indicating the presence of valid RAM data. The bus master then reads the RAM data and negates BDIN L. MRV11-BA protocol logic then responds by producing an active REC H signal, removing data from the bus, and negating BRPLY L. The bus master then responds to the passive BRPLY L signal by negating BSYNC L, terminating the bus cycle. The MRV11-BA then responds to the passive BSYNC L signal by negating RAM S H and SEL2 L signals. The passive RAM S H signal inhibits ENB H. SEL2 L (high) produces a passive (low) XMIT H signal and the RAM read sequence is completed.

RAM Write Sequence – A RAM write sequence is initiated by the addressing portion of the bus cycle as described for the RAM read sequence. However, REC H remains high for the duration of the sequence (Figure 10), enabling the receiver portions of the bus transceivers. The data portion of the sequence is initiated when the bus master device places the write data word on BDAL (0:15) L for a DATO operation, or a data byte on BDAL (0:7) L (low byte) or BDAL (8:15) L (high byte). The bus master then asserts BDOUT L, indicating that valid write data is on the bus. The MRV11-BA protocol logic responds to BDOUT L by asserting both OUT HB L and OUT LB L if BWTBT L is presently not asserted (high) by the bus master (DATO bus cycle), or only one OUT HB/LB L signal if BWTBT L is asserted (low). The logical state of BDAL0 during the addressing portion of the sequence determines which OUT signal becomes active. In this manner, BDAL0 L serves as a byte pointer. If it was not asserted (high) during the addressing portion of the sequence, OUT LB L goes active (low), enabling writing into the low byte only of the addressed RAM location; similarly, if BDAL0 L was asserted (low), OUT HB L goes low, enabling writing into the high byte only of the addressed RAM location.

The protocol logic also responds to BDOUT L by asserting BRPLY L 600 ns after receiving BDOUT L, indicating that the write operation has been completed. The bus master responds to BRPLY L by negating BDOUT L. The protocol logic then responds to the high BDOUT L signal by negating the OUT HB L and/or OUT LB L signal(s) and terminating BRPLY L.

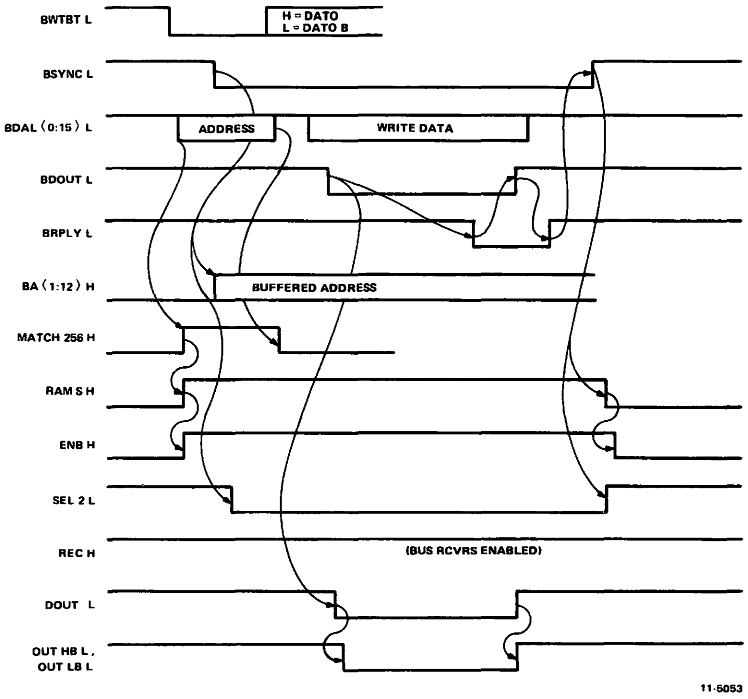


Figure MRV11-BA-10
RAM Write Sequence (DATO or DATOB)

Finally, the bus master responds to the passive (high) BRPLY L signal by negating BSYNC L and terminating the bus cycle. The MRV11-BA then responds to the passive BSYNC L signal by terminating the RAM S H, ENB H, and SEL2 L signals and the RAM write sequence is completed.

Charge Pump Circuit

The charge pump circuit produces the –5 V operating power for the PROM array integrated circuits. The basic components comprising the charge pump circuit are shown in Figure 11. Input power is obtained from the +12 V present on the LSI-11 bus. Hence, the MRV11-BA module does not require external power other than the usual +5 V and +12 V present on the backplane.

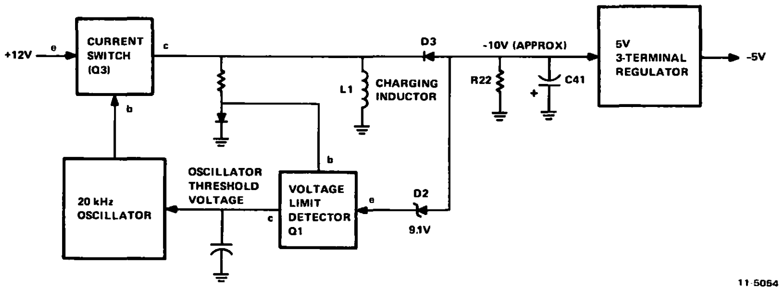


Figure MRV11-BA-11
–5 V Charge Pump Circuit (Simplified)

The oscillator provides the basic rectangular pulse that drives current switch Q3. When the oscillator turns Q3 on, +12 V is applied to L1 for approximately 25 ms and an increasing current is produced. When the oscillator turns off, the energy stored in L1 produces a negative voltage (at the top of L1 as shown in the figure), charging C41 via diode D3. Thus, stored energy in L1 is transferred to C41 as a negative voltage. Successive oscillator pulses cause C41's voltage to build up to approximately 10 V. At this point, the zener voltage of D2 is exceeded and Q1 conducts. Q1 then produces a threshold control voltage that reduces the duty cycle of the oscillator drive voltage applied to Q3 ("on" time is decreased and "off" time is increased). The feedback circuit thus produced automatically adjusts the duty cycle of the 20 kHz oscillator to control the energy stored in L1 and maintain C41's voltage at –10 V under any normal load conditions.

The actual regulated –5 V output is produced by a 3-terminal, –5 V regulator. The regulator also contains overcurrent and thermal overload protection circuits.

MRV11-C

ROM MODULE

GENERAL

The MRV11-C is a flexible, high-density ROM module used with the LSI-11 bus. The module contains sixteen 24-pin sockets which accept a variety of user-supplied ROM chips. It will accept masked ROMs, fusible link PROMs, and ultraviolet erasable PROMs. It accepts several densities of ROM chips up to and including 4K × 8 chips. Using these high-density chips gives the module a total capacity of 64K bytes.

The contents of the module can be accessed in one of two ways—either directly or window-mapped. Direct access provides total random access to all ROM locations on the module. Window-mapping provides two 2K-byte windows in memory address space to access 2K-byte segments of the ROM array. The segments that are viewed through each window can be varied under program control.

Features

- 16K, 32K, or 64K bytes of ROM
- Choice of EPROM, fusible link PROM or masked ROM
- 18-bit addressing
- Window-mapping
- Bootstrap capability

Specifications

Identification	M8048
Size	Double
Power	+5 Vdc, 0.8 A
Bus Loads	
AC	1.0
DC	1.0
ROM Specifications	
Power	+5V ±5%
Pins	24 Pin Spacing
Access Time	up to 450 ns
Size	1K × 8.2 K × 8 or 4K × 8 bits
Type	See accompanying tables for a partial listing

UV PROMs

UV PROMs	**Chip Array Size**	**Max Memory Size**
Intel 2758	1K × 8	16K bytes
Intel 2716	2K × 8	32K bytes
Intel 2732	4K × 8	64K bytes
Mostek MK2716	2K × 8	32K bytes
TI TMS 2516	2K × 8	32K bytes
TI TMS 2532	4K × 8	64K bytes

PROMs

PROM	**Chip Array Size**	**Max Memory Size**
Intel 3628	1K × 8	16K bytes
Signetics 82S 2708	1K × 8	16K bytes
Signetics 82S 181	1K × 8	16K bytes
Signetics 82S 191	2K × 8	32K bytes

CONFIGURATION

The MRV11-C Read Only Memory (ROM) contains 129 wirewrap pins and 16 ROM chip sockets. The user configures the module to the desired operating mode by installing jumper wires between the wirewrap pins. The physical location of the pins and their identification are detailed in the accompanying figure. The module is shipped from the factory with *no* jumper wires installed.

The size of the memory array is determined by the size of the ROM chips installed. The user provides these chips and inserts them into the sockets as shown in the figure. All the ROM chips must be the same array size, that is, either 1K × 8, 2K × 8, or 4K × 8 bits. The pin configuration of the chips must also be the same. The user can populate the MRV11-C for any of the three maximum memory sizes: 16K bytes, 32K bytes, or 64K bytes. Subsets of these sizes can also be chosen as shown in the table.

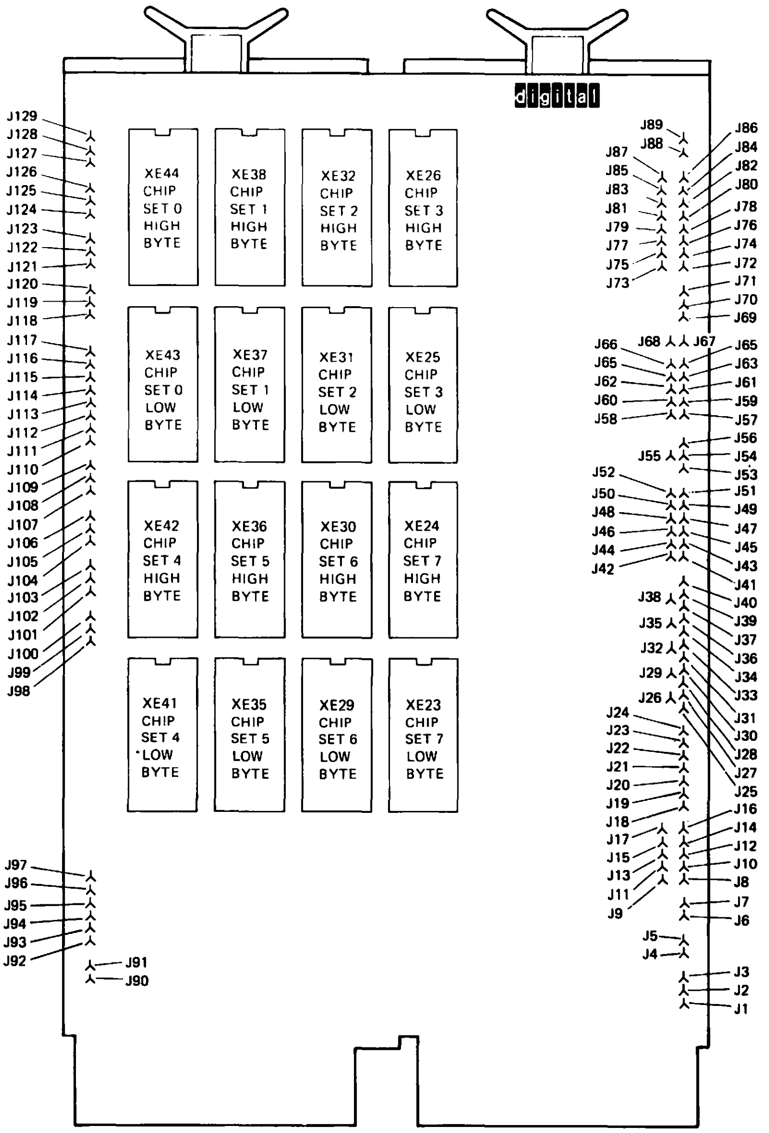

digital
XE44 CHIP SET 0 HIGH BYTE
XE38 CHIP SET 1 HIGH BYTE
XE32 CHIP SET 2 HIGH BYTE
XE26 CHIP SET 3 HIGH BYTE
XE43 CHIP SET 0 LOW BYTE
XE37 CHIP SET 1 LOW BYTE
XE31 CHIP SET 2 LOW BYTE
XE25 CHIP SET 3 LOW BYTE
XE42 CHIP SET 4 HIGH BYTE
XE36 CHIP SET 5 HIGH BYTE
XE30 CHIP SET 6 HIGH BYTE
XE24 CHIP SET 7 HIGH BYTE
XE41 CHIP SET 4 LOW BYTE
XE35 CHIP SET 5 LOW BYTE
XE29 CHIP SET 6 LOW BYTE
XE23 CHIP SET 7 LOW BYTE
J1 J2 J3 J4 J5 J6 J7 J8 J9 J10 J11 J12 J13 J14 J15 J16 J17 J18 J19 J20 J21 J22 J23 J24 J25 J26 J27 J28 J29 J30 J31 J32 J33 J34 J35 J36 J37 J38 J39 J40 J41 J42 J43 J44 J45 J46 J47 J48 J49 J50 J51 J52 J53 J54 J55 J56 J57 J58 J59 J60 J61 J62 J63 J65 J65 J66 J67 J68 J69 J70 J71 J72 J73 J74 J75 J76 J77 J78 J79 J80 J81 J82 J83 J84 J85 J86 J87 J88 J89
J90 J91 J92 J93 J94 J95 J96 J97 J98 J99 J100 J101 J102 J103 J104 J105 J106 J107 J108 J109 J110 J111 J112 J113 J114 J115 J116 J117 J118 J119 J120 J121 J122 J123 J124 J125 J126 J127 J128 J129

Figure MRV11-C-1
MRV11-C Wirewrap Pin Locations

Storage Capacity per Board as a Function of Chip Array Size and Number of Chips

Number of Chips Installed	2758 (Typ) 1024 × 8	Chip Array Size 2716 (Typ) 2048 × 8	2732 (Typ) 4096 × 8
2	2 Kb	4 Kb	8 Kb
4	4 Kb	8 Kb	16 Kb
6	6 Kb	12 Kb	24 Kb
8	8 Kb	16 Kb	32 Kb
10	10 Kb	20 Kb	40 Kb
12	12 Kb	24 Kb	48 Kb
14	14 Kb	28 Kb	56 Kb
16	16 Kb	32 Kb	64 Kb

DIRECT ADDRESS MODE

Direct Address Mode: Configuring Memory Array Size

The MRV11-C ROM module can be used in the direct addressing

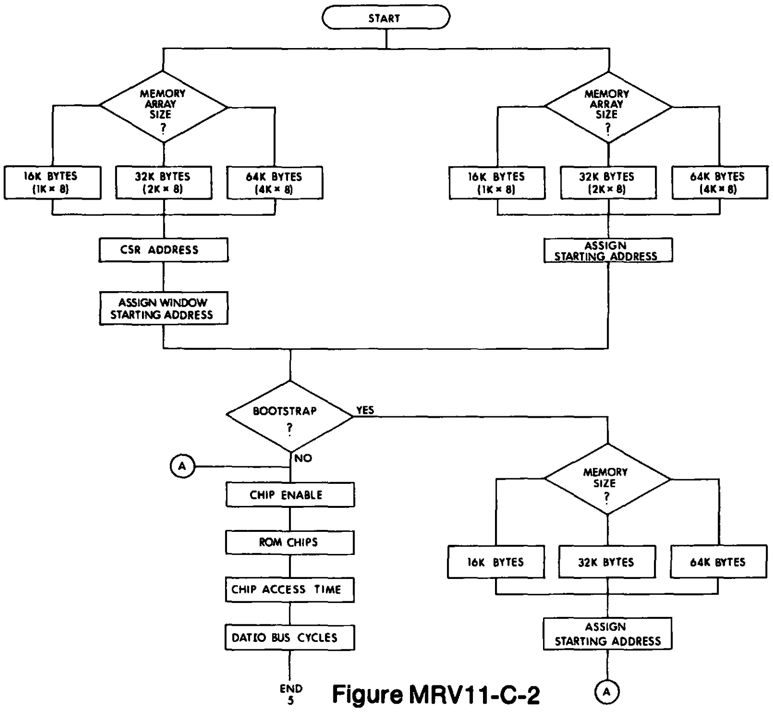


Figure MRV11-C-2
Configuration Guide

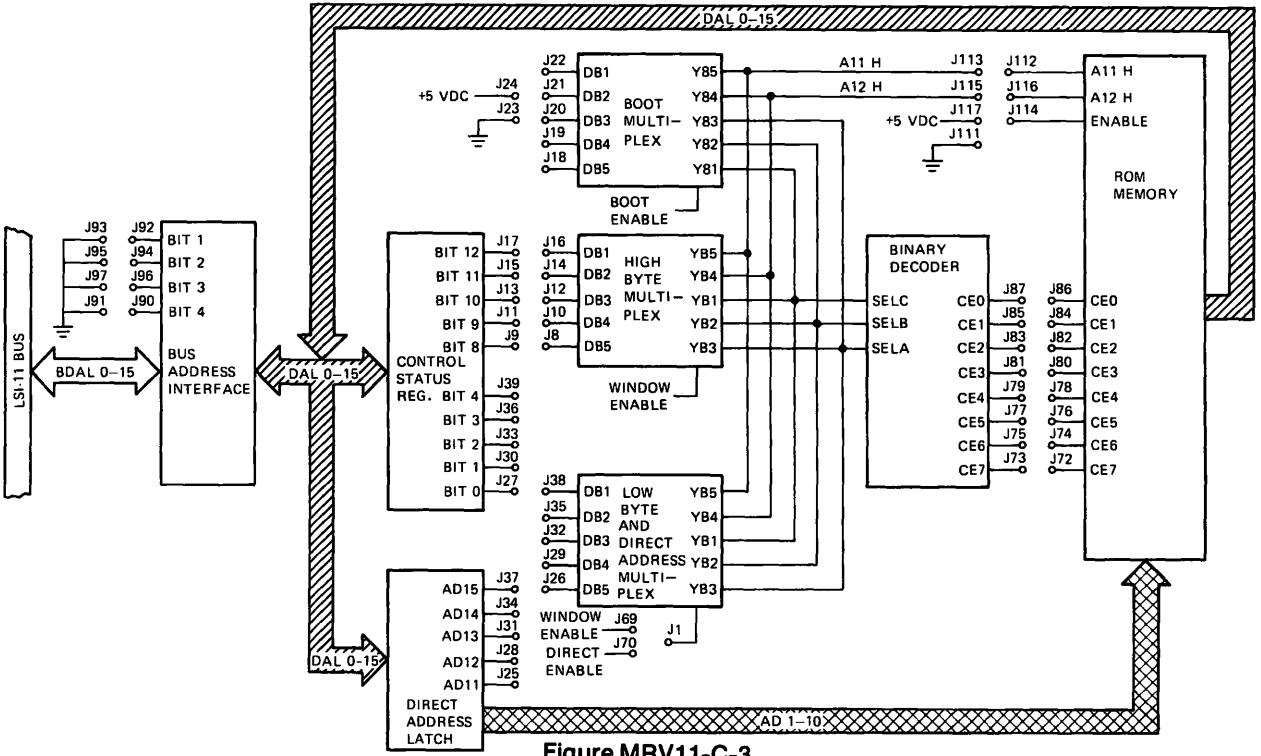


Figure MRV11-C-3
Configuration Interconnections

To configure this module, see the flowchart shown in the figure below. To avoid overlooking any jumpers, please conform to this flowchart.

The MRV11-C ROM module operates in either the direct addressing mode or the window mapping mode. The user determines the desired operating mode and must configure the module as detailed. The configuration of these two modes and the optimal bootstrap area is controlled in the module by three multiplexers, as shown in this figure: the (CSR) low byte and direct address MUX, the (CSR) high byte MUX and the boot MUX. The outputs of these multiplexers are used by the decoder to determine the proper high-order address to the read-only memory array via a set of configuration jumpers.

In the direct address mode, the high-bit address information to the decoder comes from the low byte and direct address multiplexer. A set of configuration jumpers are used to connect this MUX to the bus address lines of the Direct Address Latch.

In the window-mapped mode, the address information to the decoder comes from both the low byte and direct address multiplexer and from the high byte multiplexer. The configuration jumpers are used here to connect the multiplexers to the low byte and high byte, respectively, of the Control and Status Register (CSR). Each byte of the CSR is then used by the program to define that portion of the ROM that is accessed by the two mapping windows.

When the bootstrap is used, the address information in the decoder is provided by the boot multiplexer. The configuration jumpers here are used to define where the starting address is for the boot and to route this address information to the memory array.

Additional jumpers are provided to allow the user to tailor the MRV11-C to specific system needs, in addition to mode and memory array size. The bootstrap option can be used by either mode or can be disabled. Once the size and number of chips is determined, the chip enable jumpers can be selected. The chip access time must be selected to accommodate the chips with the slowest access time. The user can also inhibit the DATIO bus cycles, which are attempts to write into the read-only memory.

mode as part of a 16-bit or 18-bit memory system. In this mode, a jumper wire is installed between wirewrap pins J70 and J71 to enable the low byte multiplexer. To disable the window mapping mode, a jumper is installed between wirewrap pins J6 and J7. A fully populated module will be 16K bytes, 32K bytes, or 64K bytes of memory, depending on the array size as shown in the figure. Each array size requires a different configuration and the user must select the one that applies.

16K Byte Memory — The user installs up to 16 ROM chips, 1K × 8, for a maximum of 16K bytes of memory. To enable the direct addressing mode, install a jumper wire between wirewrap pins J55 and J56. The user installs jumper wires to connect the direct addressing bits to the low byte multiplexer as listed in the table.

16K Byte Direct Addressing Jumpers

Function	Jumpers Installed
Enable low-byte MUX	J70 to J71
Disable window mode	J6 to J7
Enable 16K direct mode	J55 to J56
Address bit AD11	J25 to J32
Address bit AD12	J28 to J35
Address bit AD13	J31 to J38

32K Byte Memory — The user installs up to 16 ROM chips, 2K × 8, for a maximum of 32K bytes of memory. To enable the direct addressing mode, install a jumper wire between wirewrap pins J54 and J55. The user installs jumper wires to connect the direct address bits to the low byte multiplexer as listed in the table below. In addition to those listed in the table, the A11 jumper wire must be installed between pins J112 and J113.

32K Byte Direct Addressing Jumpers

Function	Jumpers Installed
Enable low-byte MUX	J70 to J71
Disable window mode	J6 to J7
Enable 32K direct mode	J54 to J55
Chip enable input, address bit AD11	J112 to J113
Address bit AD11	J25 to J26
Address bit AD12	J28 to J32
Address bit AD13	J31 to J35
Address bit AD14	J34 to J38

64K Byte Memory — The user installs up to 16 ROM chips, 4K × 8, for a maximum of 64K bytes of memory. To enable the direct addressing mode, install a jumper wire between wirewrap pins J53 and J55. The user installs jumper wires to connect the direct address bits to the low byte multiplexer as listed in the table below. In addition to those listed in the table, the A11 and A12 jumper wires must be installed between pins J112 and J113, and J115 and J116.

64K Byte Direct Addressing Jumpers

Function	Jumpers Installed
Enable low-byte MUX	J70 to J71
Disable window mode	J6 to J7
Enable 64K direct mode	J53 to J55
Chip enable input, address bit AD11	J112 to J113
Chip enable input, address bit AD12	J115 to J116
Address bit AD11	J25 to J26
Address bit AD12	J28 to J29
Address bit AD13	J31 to J32
Address bit AD14	J34 to J35
Address bit AD15	J37 to J38

Direct Addressing Mode: Assigning the Starting Address
When the module is used in the direct addressing mode, the user determines the address space where the memory will reside. This address space is configured on 8K byte (4K word) boundaries. The range of the direct addresses required depends on the amount of memory installed on the module. The minimum is 2K bytes and the maximum is 64K bytes. Once the address space is determined, the starting address (in words) is configured by installing jumper wires. All the jumper wire configurations for the 8K byte (4K word) boundaries are listed in the table below.

The starting address and the bank of addresses assigned determines the addressing sequence of the chip sets. The figure shows examples of 32K byte and 64K byte memories and how the starting address determines which chip is accessed. The user must insert the ROM chips according to the starting address if the data is to be accessed sequentially. The 32K byte example in the figure shows that chip set 6 is the first ROM to be accessed and that chip set 5 is the last ROM to be accessed.

MRV11-C

Starting Address	Bank	Bit 17 57 to 60	Bit 16 59 to 58	Bit 15 61 to 62	Bit 14 63 to 64	Bit 13 65 to 66
0	0	I	I	I	I	I
20000	1	I	I	I	I	R
40000	2	I	I	I	R	I
60000	3	I	I	I	R	R
100000	4	I	I	R	I	I
120000	5	I	I	R	I	R
140000	6	I	I	R	R	I
160000	7	I	I	R	R	R
200000	10	I	R	I	I	I
220000	11	I	R	I	I	R
240000	12	I	R	I	R	I
260000	13	I	R	I	R	R
300000	14	I	R	R	I	I
320000	15	I	R	R	I	R
340000	16	I	R	R	R	I
360000	17	I	R	R	R	R
400000	20	R	I	I	I	I
420000	21	R	I	I	I	R
440000	22	R	I	I	R	I
460000	23	R	I	I	R	R
500000	24	R	I	R	I	I
520000	25	R	I	R	I	R
540000	26	R	I	R	R	I
560000	27	R	I	R	R	R
600000	30	R	R	I	I	I
620000	31	R	R	I	I	R
640000	32	R	R	I	R	I
660000	33	R	R	I	R	R
700000	34	R	R	R	I	I
720000	35	R	R	R	I	R
740000	36	R	R	R	R	I
760000	37	R	R	R	R	R

R = Jumper Removed
I = Jumper Installed

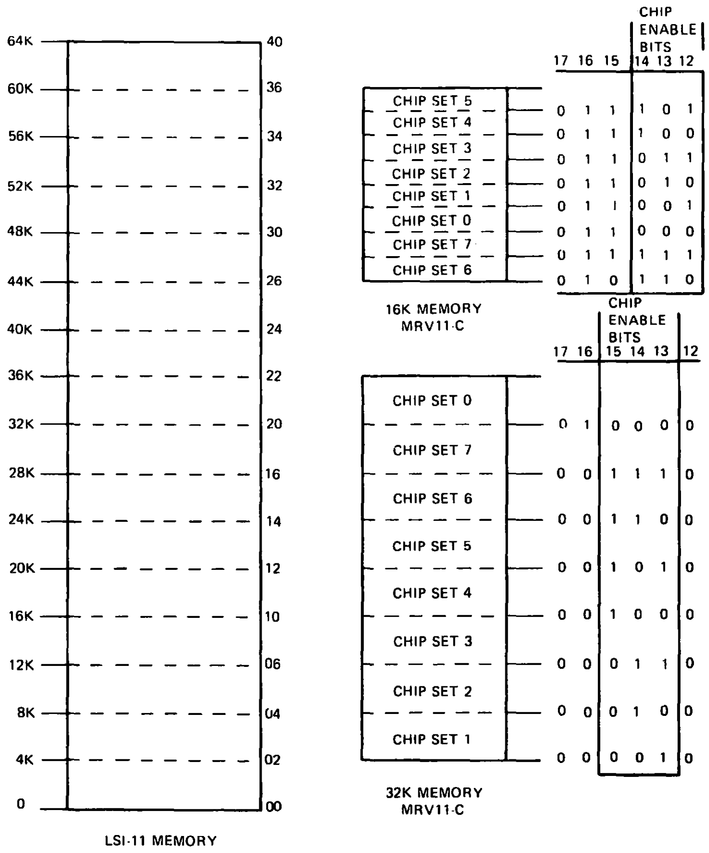


Figure MRV11-C-4
Typical MRV11-C Memory Mapping

WINDOW MAPPING MODE

Window Mapping Mode: Configuring Memory Array Size

The MRV11-C ROM module can be used in the window mapping mode as part of a 16- or 18-bit memory systems. In this mode, a jumper is installed between wirewrap pins J69 and J71 to enable the low-byte multiplexer. In addition, the user should insure that there is no jumper wire between pins J6 and J7. A fully populated module will be 16K bytes, 32K bytes or 64K bytes of memory, depending on the array size as illustrated in the figure. Each array size requires a different configuration and the user must select the one that applies.

16K Byte Memory — The user installs up to 16 ROM chips, 1K × 8, for a maximum of 16K bytes of memory. This configuration uses CSR bits 0, 1 and 2 for the low byte and CSR bits 8, 9 and 10 for the high byte. The six CSR bits are connected to the chip enable multiplexer by the jumper wires listed in the table.

16K Byte Window Mode Jumpers

CSR Output	Jumpers Installed
Low Byte	
CSR bit 0	J27 to J32
CSR bit 1	J30 to J35
CSR bit 2	J33 to J38
High Byte	
CSR bit 8	J9 to J12
CSR bit 9	J11 to J14
CSR bit 10	J13 to J16
Enable low-byte MUX	J69 to J71

32K Byte Memory — The user installs up to 16 ROM chips, 2K × 8, for a maximum of 32K bytes of memory. This configuration uses CSR bits 0, 1, 2 and 3 for the low byte and CSR bits 8, 9, 10 and 11 for the high byte. The eight CSR bits are connected to the chip enable multiplexer by the jumper wires listed in the accompanying table.

32K Byte Window Mode Jumpers

CSR Output	Jumpers Installed
Low Byte	
CSR bit 0	J27 to J26
CSR bit 1	J30 to J32
CSR bit 2	J33 to J35
CSR bit 3	J36 to J38

High Byte	
CSR bit 8	J9 to J8
CSR bit 9	J11 to J12
CSR bit 10	J13 to J14
CSR bit 11	J15 to J16
Enable low-byte MUX	J69 to J71
Address bit AD11	J112 to J113

64K Byte Memory — The user installs up to 16 ROM chips, 4K × 8, for a maximum of 64K bytes of memory. This configuration uses CSR bits 0, 1, 2, 3 and 4 for the low byte and CSR bits 8, 9, 10, 11 and 12 for the high byte. The ten CSR bits are connected to the chip enable multiplexer by the jumper wires listed in the table.

64K Byte Window Mode Jumpers

CSR Output	Jumpers Installed
Low Byte	
CSR bit 0	J27 to J26
CSR bit 1	J30 to J29
CSR bit 2	J33 to J32
CSR bit 3	J36 to J35
CSR bit 4	J39 to J38
High Byte	
CSR bit 8	J9 to J8
CSR bit 9	J11 to J10
CSR bit 10	J13 to J12
CSR bit 11	J15 to J14
CSR bit 12	J17 to J16
Enable low-byte MUX	J69 to J71
Address bit AD11	J112 to J113
Address bit AD12	J115 to J116

Window Mapping Mode: Configuring Virtual Starting Address of Windows

In the window mapping mode, the user selects a 4K byte block of address space (two consecutive 1K word segments), to be used by the window to access data from memory. Wirewrap pins J41 through J52 are used to configure the starting address of this window block, as shown in the figure below. The user selects one of the addresses and installs the jumper wires as indicated. This 4K byte block of addresses can be used by more than one MRV11-C. Refer to the Control and

Status Register description when using more than one MRV11-C. The recommended value of the window starting address is 760000. This places the windows at the bottom of the I/O page and is thus compatible with both 16-bit and 18-bit addressing. It is this recommended value that is used by the MXV11-A2 bootstrap.

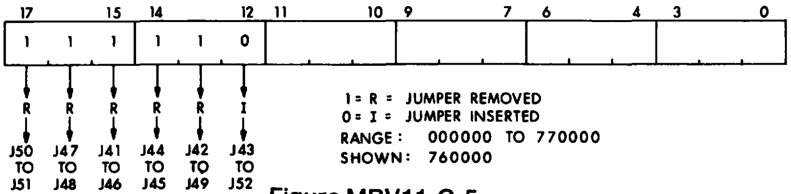


Figure MRV11-C-5
Selecting Window Starting Address

Chip Enable Function

The chip enable function decodes the contents of the high byte or the low byte of the CSR, the direct address bits or the boot address window. The decoded output will select a particular chip within the ROM. There are three multiplexers that provide inputs to the decoder. The minimum input is three bits for a 16K byte memory, up to a maximum input of five bits for a 64K byte memory. The three most significant bits of the multiplexer output are applied to the binary decoder. The other two multiplexer output bits go to the memory and are used as additional address bits. The binary input to the decoder is converted to the octal representation, and enables one of the eight chip select signals. These signals are used by the memory to enable a chip in the ROM.

I/O Control

The I/O control monitors the bus commands and provides the necessary protocol to establish communications between the MRV11-C and the bus. The DC004 chip controls the data into and out of the Control and Status Register. The BWTBT L and BDOUT L bus signals enable the MRV11-C to receive data. The BDIN L bus signal enables the MRV11-C to transmit data. Bus address bits BAD16 L and BAD17 L are buffered and only used in systems that require 18-bit addresses. The spare bus signal SSPARE 3 is used by the bus to remotely disable the MRV11-C module. The DAL15 bit is used to disable the window addressing mode by being set and a jumper wire installed between pins J67 and J68. A jumper option on the MRV11-C allows this board to respond to both DATI and DATIO bus cycles to allow use with programs using the KEV11 extended instructions on the LSI-11/2 processor.

Direct Mode Operation
When in direct mode, the MRV11-C serves as a high-density replacement for the MRV11-AA or MRV11-BA ROM modules. The base address of the direct mode ROM area is assignable on any 8K-byte boundary from 0 to 248K bytes (000000 to 760000). When operated in this mode, the application program is executed directly from the MRV11-C storage.

In this mode of operation, address bits AD11 through AD15 are used to access data in the ROM. Bits AD14 and AD15 are multiplexed and decoded to enable the memory chips. Bits AD11 and AD12 are used to determine which portion of the chip is being accessed. Only address bits AD13, AD14, and AD15 are used in 64K byte systems, but all five address bits (AD13 through AD17) are used for 256K byte systems. The starting address must be configured to start on 4K boundaries as determined by the user.

Window Mapped (Paged) Mode Operation
When window mapped operation is selected, the entire contents of the ROM board are not visible to the LSI-11 address space at any particular point in time. Instead, any two 2Kb segments of the ROM can be addressed through two independent windows defined in the LSI-11 system's address space. The association of segments of the ROM board with windows is controlled by a Control and Status Register.

The window address function uses a comparator to monitor address bits A16 and A17, and address bits DAL 12 and DAL 15. The user wires the desired address to the comparator and when the bus selects one of these addresses, the window function is enabled. This function can be disabled by installing a jumper wire across pins J6 and J7.

Bus Address Interface
The bus address interface uses four DC005 transceiver chips to decode the CSR address information received from the bus. These four chips receive the 16 BDAL (0—15) bits from the bus and route them to the MRV11-C internal bus bits DAL (0—15). During data time, the transceivers transfer data to or from the bus. The CSR is the only location that receives data from the bus. The ROM will always tranfer data to the bus.

Control and Status Register
Each MRV11-C board uses one 16-bit Control and Status Register located in the system I/O page to determine mapping of ROM segments into windows in the window mapped mode. The default address for this CSR is 177000 (777000 in the 11/23 system). The valid address range for CSRs is 177000 to 177036 (777000 to 777036 on 11/23s).

The figure below shows the bit assignments for the MRV11-C Control and Status Register.

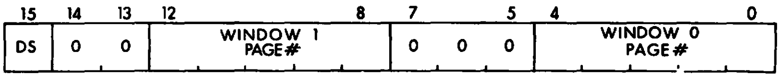


Figure MRV11-C-6
MRV11-C Control and Status Register Format

The CSR contains a 5-bit read/write field for each window. The number stored in this field (0 to 31_{10}) selects the desired 2Kb region from the MRV11-C board to be associated with the window in question. CSR bits 0 through 4 control the mapping of the low address window, window 0. The low five bits of the upper byte (bits 8 through 12) control the mapping of window 1.

The MRV11-C optionally provides a window enable/disable capability. When this option is selected, bit 15 of the CSR is used to enable or disable window response under program control. When bit 15 is a 0, the board will respond to references to the CSR or DATI or DATIO references to either of the windows. When bit 15 is a 1, only the CSR will respond. If the enable/disable option is not selected, bit 15 of the CSR will be read-only and will alway be 0. The enable/disable bit has no effect on direct mode addressing or the bootstrap window capability.

The remaining bits in the CSR (bits 5-7 and bits 13-14) are reserved and must always be zero.

Control and Status Register Addresses

CSR Address	**Bit 4 J90 to J91**	**Bit 3 J96 to J97**	**Bit 2 J94 to J95**	**Bit 1 J92 to J93**
177000	R	R	R	R
177002	R	R	R	I
177004	R	R	I	R
177006	R	R	I	I
177010	R	I	R	R
177012	R	I	R	I
177014	R	I	I	R
177016	R	I	I	I
177020	I	R	R	R

CSR Address	**Bit 4 J90 to J91**	**Bit 3 J96 to J97**	**Bit 2 J94 to J95**	**Bit 1 J92 to J93**
177022	I	R	R	I
177024	I	R	I	R
177026	I	R	I	I
177030	I	I	R	R
177032	I	I	R	I
177034	I	I	I	R
177036	I	I	I	I

R = Jumper Removed
I = Jumper Installed

NOTE
Install J67 to J68 to allow the use of bit 15 of the CSR

Window Definition
Each MRV11-C board provides a pair of 2 Kb windows. These windows are always contiguous with each other, and the base address of the window pair may be set to any 4 Kb boundary in the LSI-11 address space from 000000 to 770000. To maximize the amount of space left for system RAM, a default window base of 160000 (760000 for 11/23 systems) is suggested.

Using Multiple Boards
Up to 16 MRV11-C boards may be configured in a single system. When multiple boards are present, each board has a unique Control and Status Register address assigned in increasing order from 177000 (777000 in 11/23 systems). Each board can have a unique 4 Kb area of the physical address space set aside for its windows, but it is also possible to share one 4 Kb area of the address space among all MRV11-C boards installed in the system. This is done by using the enable/disable capability discussed earlier. When enable/disable is implemented, the disable bit in the CSR will be set automatically by BINIT on the bus or by execution of the RESET instruction. Therefore, the initial state of the system will have all boards disabled. To access a particular segment of ROM in this multi-board configuration, the programmer first enables the desired board and maps the segment. When access to that segment is completed, the board is again disabled to allow another board to be selected at a future time.

Sizing Window Mapped Boards
If the board is populated with 2048 × 8 chips, and if all 16 PROM sockets are occupied, and if the programmer selects the 16th 2Kb section, the 0th 2Kb section will be mapped by the window. If, however, a board is partially populated, an attempt to access the unpopulated area of the board will result in a bus time-out trap.

Bootstrap Window
An additional optional feature of the MRV11-C board is the capability to respond to the standard PDP-11 bootstrap addresses. When this feature is enabled, an attempt to reference addresses 173000 to 173777 (773000 to 773777 in 11/23 systems) will be automatically rerouted to a user-selected 512-byte section of th MRV11-C board. In this way, 512 bytes of the board will be visible in two places in the LSI-11 address space. The highest addressed 512 bytes of any 2 Kb segment of the board may be selected to correspond to the bootstrap window. This allows custom bootstraps to be added to an LSI-11 system without requiring an additional board to store the bootstrap.

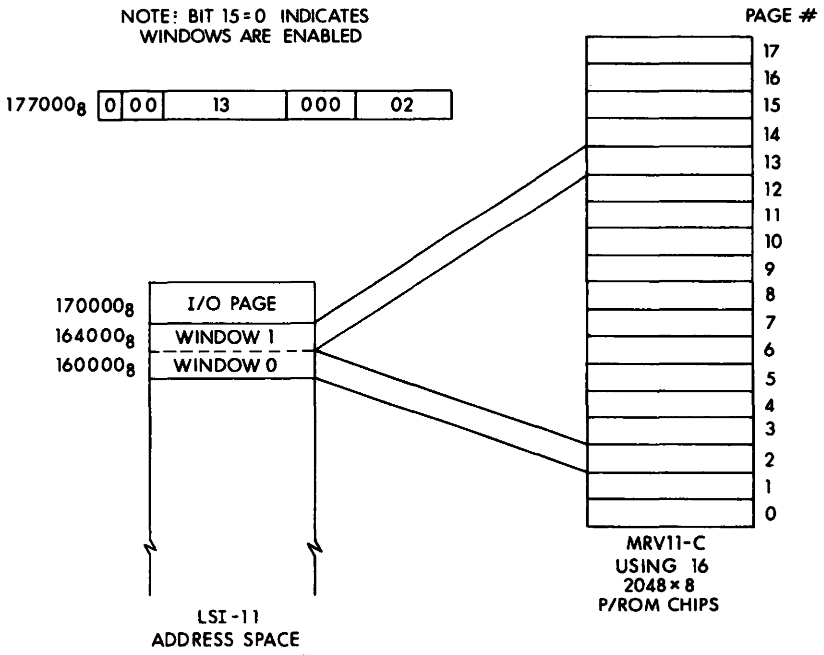


Figure MRV11-C-7
Example of Window-Mapped Operation

MULTIPLE MRV11-C MODULES

When two or more MRV11-C modules are being used in a system, the user has the option of assigning the same starting addresses or different starting addresses for the windows on each module. The window enable bit of the CSR (bit 15) is used to provide the user software control over the windows. Setting bit 15 to a 1 disables both windows on the respective MRV11-C. In order to use bit 15 of the CSR, a jumper must be installed between pins J67 and J68 on all MRV11-C modules that are required to be disabled under software control, such as modules configured with the same window starting addresses. With the jumper installed, bit 15 will also be set upon system initialization so that the module will be disabled on power-up.

BOOTSTRAP

The MRV11-C allows the user to install a bootstrap program of up to 512 bytes. The bootstrap starting address is hardwired for 16-bit systems at 173000 and for 18-bit systems at 773000. The bootstrap program is normally enabled and must be disabled if it is not being used. To disabled the bootstrap, install a jumper wire between wirewrap pins J88 and J89. The bootstrap program is inserted as the top 512 bytes of any 2K byte page of ROM. The user installs jumper wires for the boot multiplexer (see the accompanying figure), to select the starting address for the particular page in which the bootstrap resides. The number of pages vary by the array size of the ROM chips used.

16K Byte Memory — This configuration allows the user to select any one of eight starting addresses for the bootstrap program, as shown in the table below.

Bootstrap Starting Address: Jumper Configurations for 16K Byte ROM Memory

Starting	Install Jumper Wire From		
Address	J22 to	J21 to	J20 to
003000	J24	J24	J24
007000	J24	J24	J23
013000	J24	J23	J24
017000	J24	J23	J23
023000	J23	J24	J24
027000	J23	J24	J23
033000	J23	J23	J24
037000	J23	J23	J23

NOTE

Logic 1 = J23
Logic 0 = J24
Bootstrap starting address is normalized to memory location 000000.

32K Byte Memory — This configuration allows the user to select any one of sixteen starting addresses for the bootstrap program as shown in the table below.

Bootstrap Starting Address: Jumper Configurations for 32K Byte ROM Memory

Starting	**Install Jumper Wire From**			
Address	**J22 to**	**J21 to**	**J20 to**	**J18 to**
003000	J24	J24	J24	J24
007000	J24	J24	J24	J23
013000	J24	J24	J23	J24
017000	J24	J24	J23	J23
023000	J24	J23	J24	J24
027000	J24	J23	J24	J23
033000	J24	J23	J23	J24
037000	J24	J23	J23	J23
043000	J23	J24	J24	J24
047000	J23	J24	J24	J23
053000	J23	J24	J23	J24
057000	J23	J24	J23	J23
063000	J23	J23	J24	J24
067000	J23	J23	J24	J23
073000	J23	J23	J23	J24
077000	J23	J23	J23	J23

NOTE

Logic 1 = J23
Logic 0 = J24
Bootstrap starting address is normalized to memory location 000000.

MRV11-C

64K Byte Memory — This configuration allows the user to select any one of 32 pages to install the bootstrap program. The jumper wire configurations for all 32 pages are listed in the table below and the user selects only one of these.

Bootstrap Starting Address Jumper Configuration for 64K Byte ROM Memory

Starting Address	Install Jumper Wire From				
	J22 to	J21 to	J20 to	J19 to	J18 to
003000	J24	J24	J24	J24	J24
007000	J24	J24	J24	J24	J23
013000	J24	J24	J24	J23	J24
017000	J24	J24	J24	J23	J23
023000	J24	J24	J23	J24	J24
027000	J24	J24	J23	J24	J23
033000	J24	J24	J23	J23	J24
037000	J24	J24	J23	J23	J23
043000	J24	J23	J24	J24	J24
047000	J24	J23	J24	J24	J23
053000	J24	J23	J24	J23	J24
057000	J24	J23	J24	J23	J23
063000	J24	J23	J23	J24	J24
067000	J24	J23	J23	J24	J23
073000	J24	J23	J23	J23	J24
077000	J24	J23	J23	J23	J23
103000	J23	J24	J24	J24	J24
107000	J23	J24	J24	J24	J23
113000	J23	J24	J24	J23	J24
117000	J23	J24	J24	J23	J23
123000	J23	J24	J23	J24	J24
127000	J23	J24	J23	J24	J23
133000	J23	J24	J23	J23	J24
137000	J23	J24	J23	J23	J23
143000	J23	J23	J24	J24	J24
147000	J23	J23	J24	J24	J23
153000	J23	J23	J24	J23	J24
157000	J23	J23	J24	J23	J23
163000	J23	J23	J23	J24	J24
167000	J23	J23	J23	J24	J23
173000	J23	J23	J23	J23	J24
177000	J23	J23	J23	J23	J23

NOTE
Logic 1 = J23
Logic 0 = J24

CHIP ENABLE

There are 16 sockets on the MRV11-C module available for ROM chips. If the module is not fully populated, then the chip enable signals for the sockets without ROMs should not be jumpered. This prevents the program from accidently addressing the sockets in which there are no ROMs. It is recommended that the ROMs be installed in pairs of high and low bytes. When a complete set of ROMs is installed, then all the chip enable jumper wires are installed as listed in the table below.

Chip Enable Jumpers

Sockets Enabled	Chip Enable Signal	Wirewrap Pins Jumpered
XE43, XE44	CE0	J86 to 87
XE37, XE38	CE1	J84 to J85
XE31, XE32	CE2	J82 to J83
XE25, XE26	CE3	J80 to J81
XE41, XE42	CE4	J78 to J79
XE35, XE36	CE5	J76 to J77
XE29, XE30	CE6	J74 to J75
XE23, XE24	CE7	J72 to J73

ROM CHIPS

The ROM is provided by the user and consists of up to 16 chips that are inserted into prewired sockets. The chips will be either 1K × 8 bit, 2K × 8 bit, or 4K × 8 bit ROMs. When the MRV11-C is fully populated, the result will be either 16K, 32K or 64K bytes of memory. These ROMs can be supplied by a variety of vendors and the basic configuration for many of the ROMs is standardized except for pins 18, 19, 20, and 21. The configuration of these pins will vary depending upon the size of the ROM and the vendor who supplies them. Therefore the user should verify the vendor's specifications in order to determine if a particular ROM can be used on the MRV11-C.

The MRV11-C module is configured so that the user can select the signals that are applicable to pins 18, 19, and 21. The board provides wirewrap pins for the user to select the A11, A12, 5 Vdc or ground. There are three individuals loops that interconnect all chips and three wirewrap pins available for each individual chip. Wirewrap pin J112 interconnects pin 19 of all the chips and pin J116 interconnects pin 21 of all the chips; these are normally designated as the A10 or A11 inputs to the chips. Wirewrap pin J114 interconnects wirewrap pins that are individually associated with each chip. Pin 18 of each chip is individually wired to a wirewrap pin and chip pin 20 is wired to the Chip Enable signal. Chip pin 20 is also individually wired to a wirewrap pin. The user must determine from the vendor's specifications which signals apply to which pins and must install jumper wires as needed to configure an operational module.

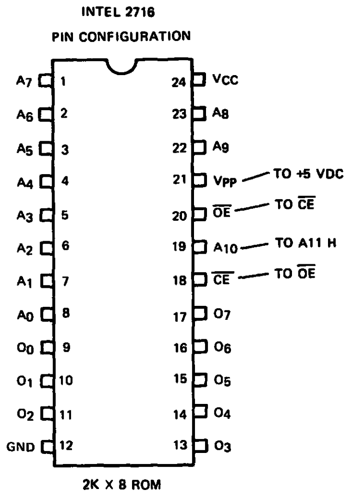
For example, in the figure below, there are pin configurations for two types of chips, a 2K × 8 ROM that is used for 32K byte memories and a 4K × 8 ROM that is used for 64K byte memories. To configure the 32K byte ROM memory, pin 19 is designated as A10 and by inserting a jumper wire between pins J112 and J113, pin 19 of all the chips is connected to A11 which is used as the A10 input. The V_{PP} input, designated by pin 21, is specified that it must be connected to a +5 Vdc source. This is accomplished by inserting a jumper wire between pins J116 and J117, which will connect pin 21 of all the chips to +5 Vdc. The OE (pin 20) and CE (pin 18) should be connected together to the Chip Enable. Therefore each chip must be connected individually and jumper wires are installed between the following pins to operate as a 32K byte memory.

J118	to	J120
J121	to	J123
J124	to	J126
J127	to	J129
J101	to	J103
J98	to	J100
J104	to	J106
J107	to	J109

Using the 4K × 8 ROM to configure a 64K byte memory, pin 19 is designated as A10 and by inserting a jumper wire between pins J112 and J113, pin 19 of all the chips is connected to A11 which is used as the A10 input. However, pin 21 is now designated as A11 and this must be connected to A12. This is accomplished by inserting a jumper wire between pins J116 and J115, which will connect pin 21 of all the chips to A12. The OE/Vpp (pin 20) and CE (pin 18) should be connected together to the Chip Enable. Therefore each chip mut be connected individually and jumper wires are installed between the following pins to operate as a 64K byte memory.

J118	to	J120
J121	to	J123
J124	to	J126

J127	to	J129
J101	to	J103
J98	to	J100
J104	to	J106
J107	to	J109



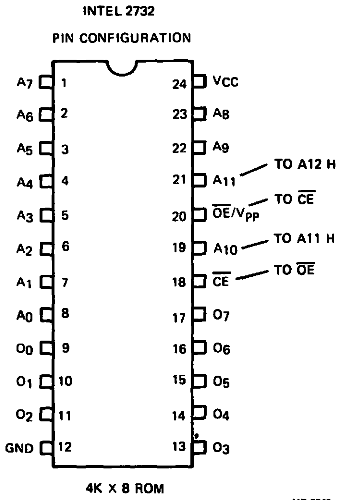


Figure MRV11-C-8
Pin Configuration

Chip Access Time

The MRV11-C can normally interface with chips that have an access time of less than 50 ns. The chip access time is determined by the slowest access time of any individual chip installed on the MRV11-C. If the chip access time is greater than 50 ns and less than 200 ns, then an RC delay can be incorporated into the circuits. This is done by installing jumper wires between wirewrap pins J1 and J3 and pins J2 and J3. If the chip access time is greater than 200 ns and less than or equal to 450 ns, the jumper wire between wirewrap pins J1 and J3 is removed and the jumper wire between wirewrap pins J2 and J3 remains inserted.

DATIO Bus Cycle

The processor may attempt to perform DATIO bus cycles to the MRV11-C. These bus cycles are attempts to write the data into the memory (which is read-only memory). The only write operation needed by the MRV11-C involves writing data into the Control Status Register (CSR). All other attempts are timed out by the bus cycle. This condition is allowed unless a jumper wire is installed between wirewrap pins J4 and J5. With this jumper installed, the BDOUT is inhibited except when the bus is addressing the CSR. This eliminates any writing attempts from the bus except those for the Control Status Register. The MRV11-C normally responds to DATIO bus cycles and installing the jumper will cause a timeout for a DATIO bus cycle to the ROM.

Wirewrap Pin Identification

The MRV11-C module provides the user with 129 wirewrap pins to configure the module for many types of applications. These wirewrap pins are identified and located on the module in the figure at the beginning of the chapter. In the table below, the wirewrap pins are numerically listed with descriptions of their functional use.

Wirewrap Pin Identification

Wirewrap Pin Designation	Function
J1	RXCX pull-up resistor
J2	RXCX optional capacitor
J3	RXCX signal
J4	LMATCH input for BDOUT control
J5	LMATCH for BDOUT control

Wirewrap Pin Designation	Function
J6	Window address enable ground
J7	Window address enable
J8	High byte chip enable bit A11
J9	CSR high byte bit 8 chip enable output
J10	High byte chip enable bit A12
J11	CSR high byte bit 9 chip enable output
J12	High byte chip enable least significant bit
J13	CSR high byte bit 10 chip enable output
J14	High byte chip enable intermediate bit
J15	CSR high byte bit 11 chip enable output
J16	High byte chip enable most significant bit
J17	CSR high byte bit 12 chip enable output
J18	Boot address chip enable bit A11
J19	Boot address chip enable bit A12
J20	Boot address chip enable least significant bit
J21	Boot address chip enable intermediate bit
J22	Boot address chip enable most significant bit
J23	Boot address chip enable logic 1
J24	Boot address chip enable logic 0
J25	Direct address bit 11 chip enable output
J26	Low byte chip enable A11 H bit
J27	CSR low byte bit 0 chip enable output
J28	Direct address bit 12 chip enable output
J29	Low byte chip enable A12 bit
J30	CSR low byte bit 1 chip enable output
J31	Direct address bit 13 chip enable output
J32	Low byte chip enable least significant bit
J33	CSR low byte bit 2 chip enable output

Wirewrap Pin Designation	Function
J34	Direct address bit 14 chip enable output
J35	Low byte chip enable intermediate bit
J36	CSR low byte bit 3 chip enable output
J37	Direct address bit 15 chip enable output
J38	Low byte chip enable most significant bit output
J39	CSR low byte bit 4 chip enable output
J40	Not used (Reserved for future DIGITAL use)
J41	Window address bit 15 compare ground
J42	Window address bit 13 compare input
J43	Window address bit 12 compare ground
J44	Window address bit 14 compare input
J45	Window address bit 14 compare ground
J46	Window address bit 15 compare input
J47	Window address bit 16 compare ground
J48	Window address bit 16 compare input
J49	Window address bit 13 compare ground
J50	Window address bit 17 compare input
J51	Window address bit 17 compare ground
J52	Window address bit 12 compare input
J53	Direct address 32K memory limit output
J54	Direct address 16K memory limit output
J55	Direct address memory limit input
J56	Direct address 8K memory limit output
J57	Direct address bit 17 compare ground
J58	Direct address bit 16 compare input
J59	Direct address bit 16 compare ground
J60	Direct address bit 17 compare input
J61	Direct address bit 15 compare ground

Wirewrap Pin Designation	Function
J62	Direct address bit 15 compare input
J63	Direct address bit 14 compare ground
J64	Direct address bit 14 compare input
J65	Direct address bit 13 compare ground
J66	Direct address bit 13 compare input
J67	CSR high byte bit 15 enable ground
J68	CSR high byte bit 15 enable input
J69	High byte chip enable window address function
J70	High byte chip enable direct address function
J71	High byte chip enable function select drivers
J72	Bit 7 chip select enable input
J73	Bit 7 chip enable decoder output
J74	Bit 6 chip select enable input
J75	Bit 6 chip enable decoder input
J76	Bit 5 chip select enable input
J77	Bit 5 chip enable decoder output
J78	Bit 4 chip select enable input
J79	Bit 4 chip enable decoder output
J80	Bit 3 chip select enable input
J81	Bit 3 chip enable decoder output
J82	Bit 2 chip select enable input
J83	Bit 2 chip enable decoder output
J84	Bit 1 chip select enable input
J85	Bit 1 chip enable decoder output
J86	Bit 0 chip select enable input
J87	Bit 0 chip enable decoder output
J88	Boot address enable ground
J89	Boot address enable

Wirewrap Pin Designation	**Function**
J90	DAL 4 CSR address select signal
J91	DAL 4 CSR address select ground
J92	DAL 1 CSR address select signal
J93	DAL 1 CSR address select ground
J94	DAL 2 CSR address select signal
J95	DAL 2 CSR address select ground
J96	DAL 3 CSR address select signal
J97	DAL 3 CSR address select ground
J98	Pin 18 input for chip set 5
J99	Chip wirewrap interconnection for chip set 5
J100	Pin 20 input for chip set 5 (Chip Enable 5)
J101	Pin 18 input for chip set 4
J102	Chip wirewrap interconnection for chip set 4
J103	Pin 20 input for chip set 4 (Chip Enable 4)
J104	Pin 18 input for chip set 6
J105	Chip wirewrap interconnection for chip set 6
J106	Pin 20 input for chip set 6 (Chip Enable 6)
J107	Pin 18 input for chip set 7
J108	Chip wirewrap interconnection for chip set 7
J109	Pin 20 input for chip set 7 (Chip Enable 7)
J110	Not used (Reserved for future DIGITAL use)
J111	ROM interconnection, ground reference
J112	Chip enable bit bus input
J113	Address bit A11, used as chip input A10
J114	Chip interconnection loop (to wirewrap pins)
J115	Address bit A12, used as chip input A11
J116	Chip interconnection loop for chip pin 21
J117	ROM interconnection for chip set 0

Wirewrap Pin Designation	Function
J118	Pin 18 input for chip set 0
J119	Chip wirewrap interconnection for chip set 0
J120	Pin 20 input for chip set 0 (Chip Enable 0)
J121	Pin 18 input for chip set 1
J122	Chip wirewrap interconnection for chip set 1
J123	Pin 20 input for chip set 1 (Chip Enable 1)
J124	Pin 18 input for chip set 2
J125	Chip wirewrap interconnection for chip set 2
J126	Pin 20 input for chip set 2 (Chip Enable 2)
J127	Pin 18 input for chip set 3
J128	Chip wirewrap interconnection for chip set 3
J129	Pin 20 input for chip set 3 (Chip Enable 3)

PROGRAMMING

Window Mapped Mode Application Examples

The window mapped mode may be used in two different ways in LSI-11 application programs. The application can be coded in such a way as to execute directly from the windows, or the window mapped board may be used as a program load device to transfer a stand-alone application program from ROM into RAM memory at system start-up.

Executing Windowed Programs — Executing directly from MRV11-C windows allows very large program sizes, up to 56K Bytes of RAM on LSI-11/2 systems. However, software to be executed in this mode must be custom-designed and must be written in assembly language.

An application designed for windowed execution must have a mechanism for calling a subroutine or transferring control to another routine which is physically located in a presently unmapped section of the windowed ROM board. To accomplish this, we must use a technique different from the standard JSR or JMP instructions. A method for doing this is illustrated below. The routine which processes subroutine calls and jumps to other pages must, of course, be located in a section of memory which is not window mapped. To call a subroutine using these capabilities, one would write CALLW0, label rather than JSR PC, label. This would cause the subroutine desired to be mapped into window 0 and the call to be executed. Upon subroutine return, which is done with a normal RTS PC instruction, the original mapping would be restored and control would be returned to the calling program unit. Likewise, to invoke a subroutine but have it mapped in window 1, the programmer codes CALLW1, label. Note that the mechanism shown below preserves condition codes from the called routine back to the caller (i.e., routines can return status in the condition codes). Instead of the unconditional jump instruction, the programmer codes JMPW0, label to jump to a routine, mapping it into window 0. Similarly, one can code JMPW1, label to transfer control to a routine which should be mapped into window 1.

To make use of this, the program should be assembled with .ENABL AMA to force absolute addressing in the assembly. At start-up time, a boot routine must be executed (from the MRV11-C boot window or elsewhere) which copies the trap handler routine to RAM memory, if necessary, and initializes the trap vector to contain the address of the trap handling routine and a new status word of all 0's.

```
W0BASE = 160000
W1BASE = 164000
JMPW = 1
JSRW = 0
W1 = 2
W0 = 0
        .MACRO      CALLW0 ADRS
        TRAP        JSRW + W0 + <<ADRS/1000> & 174>
        .WORD       W0BASE + <ADRS & 3777>
        .ENDM       CALLW0

        .MACRO      JMPW0 ADRS
        TRAP        JMPW + W0 + <<ADRS/1000> & 174>
        .WORD       W0BASE + <ADRS & 3777>
        .ENDM       JMPW0

        .MACRO      CALLW1 ADRS
        TRAP        JSRW + W1 <<ADDRS/1000> & 174>
        .WORD       W1BASE + <ADRS & 3777>
        .ENDM       JMPW1

        .MACRO      JMPW1 ADRS
        TRAP        JMPW + W1 + <<ADRS/1000> & 174>
        .WORD       W1BASE + <ADRS & 3777>
        .ENDM       JMPW1
```

```
TRPHAN:   MOV   @#MRVCSR,-(SP)   ;Save previous mapping
          TST   -(SP)            ;Reserve space for adrs
          MOV   R0, -(SP)        ;Save caller's register
          MOV   6 (SP), R0       ;And set R0 to address of
                                   TRAP + 2
          ADD   #2, 6 (SP)       ;Update return PC beyond adrs
          MOV   @R0, 2(SP)       ;Move adrs (follows
                                   TRAP instruction)
          MOV   -(R0), R0        ;R0 TRAP instruction itself
          BIC   #177600, R0      ;Extract page #, window #,
                                   JMP/JSR
          ASR   R0               ;Move JMP/JSR to C bit
          ROR   R0               ;Place window # in C,
                                   JMP/JSR in bit 15
          MOV   #MRVCSR, -(SP)   ;Set address of window
                                   0 in map bits
          ADC   @SP              ;And update based on window #
          MOVB  R0, @(SP)+       ;Map new page in
                                   selected window
          ROL   R0               JMP/JSR back to C bit
          MOV   (SP)+, R0        Restore caller's register
          BCS   1$               ;If JMP, branch to 1$
          JSR   PC, @(SP)+       ;Else JSR to desired routine
          MFPS  4(SP)            ;Store returned condition codes
                                   in old PS
          MOV   (SP)+, @#MRVCSR  ;Restore original mapping
          RTI                    ;And return after TRAP and adrs
1$        MOV   (SP)+, @SP       ;If JMP, move adrs
          MOv   (SP)+, @SP       ;Up over old (caller's) PC
          RTI                    ;And go to new location
```

JSR and JMP Window Mapped Control Routines

The programmer of this type of application must take care not to cross page boundaries without remapping to the next page. If a page boundary is encountered, the JMPW0 or JMPW1 pseudo instructions should be used to get to the beginning of the next page.

Using Window Mapping as a Program Loader — The MRV11-C in window mapped mode can also be used as a low-cost program load device for stand-alone applications. This allows application programs which cannot be easily segmented into ROM and RAM sections to be loaded from a ROM environment into RAM for execution. To use the

MRV-11C in this mode, a bootstrap loader program must be written to copy the contents of the ROM board into the RAM area at power-up. The figure below demonstrates such a program, designed to load stand-alone images which have been created by the RT-11 LINK utility. It is also possible to load an RSX-11S system image from one or more MRV11-C boards into RAM for execution.

```
MRVCSR = 177000
MRVWIN = 160000
ONEKW = 003777

LOADER:  CLR  @#MRVCSR          ;Enable & map low 1K words
         MOV  @#MRVWIN+50, R5   ;R5=RT-11 SAV file high limit
         CLR  R4                ;Start copying into location 0
1$       MOV  #MRVWIN, R3       ;Reset to base of first window
2$       MOV  (R3)+, (R4)+      ;Copy one word into RAM
         CMP  R4, R5            ;Moved highest word in program?
         BHIS 3$                ;If HIS, yes
         BIT  #ONEKW, R3        ;Have reached next 1Kw
                                   boundary?
         BNE  2$                ;If NE, no
         INC  @#MRVCSR          ;Else map next 1K in window 0
         BR   1$                ;And continue copying

3$       MOV  @#40,PC           ;Start at user's transfer address
```

Bootstrap Loader for Stand-Alone Programs in RT-11 .SAV Format

MSV11-B 4K BY 16-BIT MOS READ/WRITE MEMORY

GENERAL

The MSV11-B is a 4K by 16-bit dynamic MOS read/write memory module which can be used for storage of user programs and data. The storage capacity is 4096 16-bit words. Memory address selection is user-configured by installing or removing jumpers contained on the module.

Memory refresh is directly controlled externally by LSI-11 bus signals. The MSV11-B is LSI-11 bus-compatible and capable of either programmed I/O data transfers with the processor or DMA transfers with other LSI-11 bus modules.

FEATURES

- 4096 by 16-bit word
- Fast access time – 550 ns maximum
- Lower power – 12.7 W for the module, maximum
- Dynamic MOS memory chips – Refresh is externally controlled
- User-configured 4K addresses – Three jumpers allow user address configuration.

SPECIFICATIONS

Identification	M7944
Size	Double
Power	+5 V ± 5% at 0.6 A
	+12 V ± 3% at 0.54 A
Bus Loads	
AC	1.9
DC	1.0

CONFIGURATION

General

The user can select the 4K address space (bank) in which the module is addressed by installing or removing jumpers. The MSV11-B module is factory-configured to respond to addresses in bank 0 (addresses 0–17776) and not reply to refresh.

Address Jumpers

MSV11-B address jumpers are located as shown in Figure 1. The module is supplied with all address jumpers installed. Figure 2 illustrates a 16-bit address and how jumpers are assigned for the MSV11-B module.

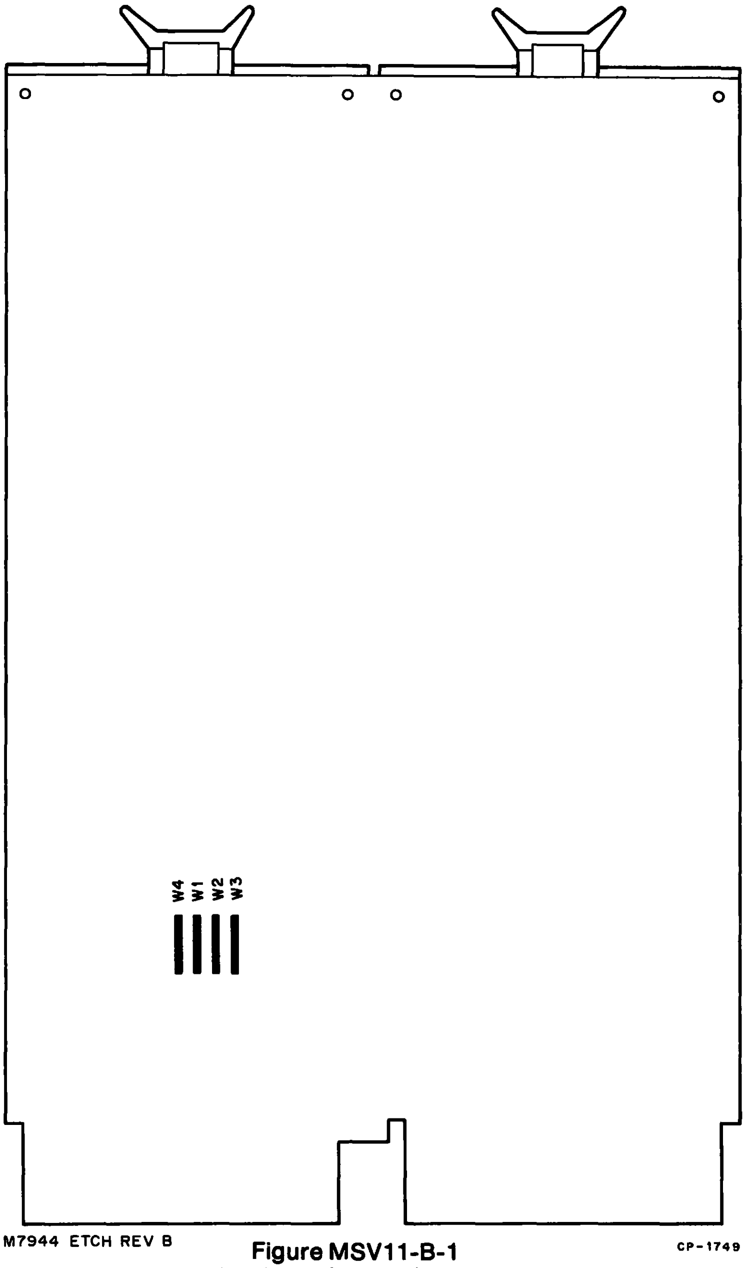
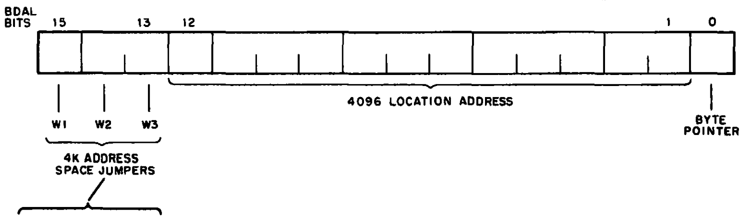


Figure MSV11-B-1
MSV11-B Jumper Locations



W1	W2	W3	Bank No.	Address Range	Octal Address Range
I	I	I	0	0-4K	000000-017776
I	I	R	1	4-8K	020000-037776
I	R	I	2	8-12K	040000-057776
I	R	R	3	12-16K	060000-077776
R	I	I	4	16-20K	100000-117776
R	I	R	5	20-24K	120000-137776
R	R	I	6	24-28K	140000-157776
R	R	R	7	28-32K	160000-177776

NOTE: I = Installed, R = Removed

CP-1883

Figure MSV11-B-2
MSV11-B Address Format/Jumpers

Reply to Refresh Jumpers
Only one dynamic memory module in a system is required to reply to the refresh bus transactions initiated by the refreshing device. The module selected to reply should be the module with the slowest access time or physically the farthest from the refreshing device. Jumper W4 enables or inhibits the MSV11-B reply as follows.

W4 installed: MSV11-B will not assert BRPLY in response to refresh bus signals.

W4 removed: MSV11-B will reply to refresh bus BSYNC/BDIN transactions by asserting BRPLY L.

Refresh Requirements
The MSV11-B module contains dynamic MOS integrated memory circuits which must be completely refreshed once every 2 ms or less. The refresh operation can be performed by the processor, by the REV11 series refresh/bootstrap module, or by special DMA logic provided by the user.

MSV11-B

FUNCTIONAL DESCRIPTION

General
Major functions contained on the MSV11-B module are shown in Figure 3. Memory data can be stored (written) or read by the processor or other bus master devices operating in the DMA mode, with appropriate bus cycles: DATO (16-word write operation); DATOB (8-bit byte write operation); DATI (16-bit read operation); or DATIOB [16-bit read-modify-write (8- or 16-bit) operation].

Addressing is initiated by a master device (either the processor or a DMA device) by placing the 16-bit address on BDAL0–15 L and asserting BSYNC L, latching the address (and bank select information) in the address register. Address bits are routed from the BDAL bus receivers onto the module's DAL0–15 H bus to the 13-bit address and bank select register input. Address bits BDAL13–15 L are decoded by the bank address decoder. Decoded output signal BS H will go active (high) only when the jumper-selected bank address is decoded. The active BS H signal is stored along with the 13-bit memory address for the duration of the operation.

The memory array comprises sixteen 16-pin 4K by 1-bit memory chips which require the address multiplexer to address the array with two 16-bit bytes. Address multiplexer control logic responds to the active SYNC H and stored active bank select (LBS L) signal by immediately generating an active row address strobe (RAS). This signal remains active for the duration of the active SYNC H signal. Address multiplex control AMX L is initially passive (high), multiplexing the stored row address bits (LDAL7–12 H) through the 12:6-bit address multiplexer and into all memory chips. After approximately 150 ns, address multiplex control logic generates an active column address strobe (CAS) and an active AMX L signal. Multiplexer column address bits (LDAL1–6 H) are then strobed into all memory chips. This completes the chip addressing portion of the memory operation.

When in a memory read operation, the bus master device asserts BDIN L. The data from the accessed memory location is present on the D0–15 H bus and at bus driver inputs. Reply logic responds to BDIN L by generating an active DRIVE L signal which gates the memory read data onto BDAL0–15 L for input to the requesting device; reply logic also asserts BRPLY L to complete the data transfer portion of the cycle.

When in a memory write operation (or the write portion of a DATIOB cycle), the addressing portion of the operation is similar to the read cycle addressing. After the addressing portion of the cycle has been completed, the master device asserts BDOUT L, and BWTBT L either goes passive (high) if a DATO (word) write cycle is to be performed, or remains asserted if a DATOB (byte) write cycle is to be performed.

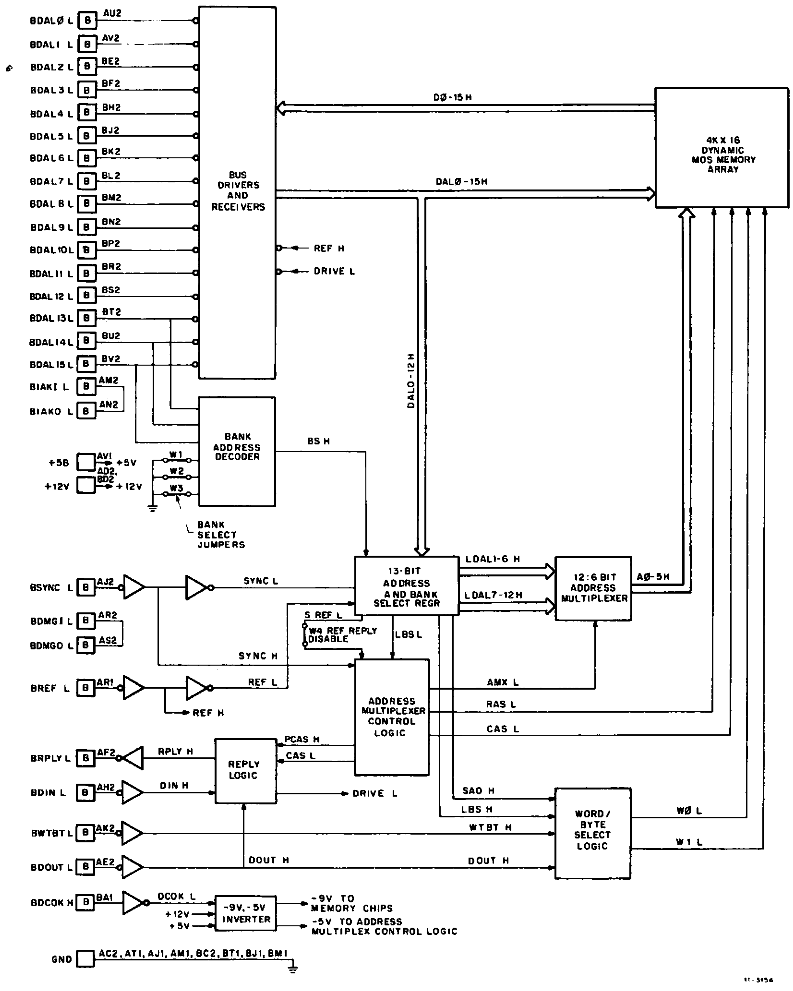


Figure MSV11-B-3
MSV11-B Logic Block Diagram

MSV11-B

Word/byte select logic responds to the DATO cycle by asserting both W0 L and W1 L for the duration of the cycle, enabling DAL0–15 H bits to be written into the address location in all memory chips. However, in a DATOB cycle with A0 H low (even byte), only W0 L goes active, enabling the writing of DAL0–7 H into the addressed location in the appropriate eight memory chips. Similarly, if A0 H is high (odd byte), only W1 L goes active, enabling only DAL8–15 H bits to be written into the addressed location in the appropriate eight memory chips. The reply logic also responds to the active BDOUT L signal by asserting BRPLY L, indicating that the data has been written, completing the data transfer.

The memory chips in the MSV11-B require a refresh operation once every 1.6 ms. This operation is entirely under the control of either processor microcode or a DMA device, as selected by the user. The address multiplex control logic responds to BREF L, generated by the refresh-controlling device, by simulating a "bank selected" operation. (All system memory banks are simultaneously selected during refresh.) Refresh is then accomplished by a device by executing 64 successive BSYNC L/BDIN L operations while incrementing BDAL1–6 by one on each bus transaction. Refresh is simply a series of forced memory read operations where only the row addresses are significant. Each of the 64 rows in all dynamic MOS memory chips in an LSI-11 system are simultaneously refreshed in this manner. The REF H signal inhibits all BDAL bus drivers for the duration of the refresh operation.

A dc-to-dc inverter circuit is included on the module for negative voltage generation. Output voltages include –9 V for the MOS memory chips and –5 V for linear devices in the address multiplex control logic. Hence, only +12 V and +5 V power inputs are required for module operation. The BDCOK H signal starts dc-to-dc inverter oscillation when bus power is applied.

MSV11-CD 16K BY 16-BIT MOS READ-WRITE MEMORY

GENERAL

The MSV11-CD is a 16K dynamic MOS read/write memory option that can be installed in any LSI-11 bus. Memory contents are volatile; that is, when operating power is removed, memory data is lost.

FEATURES

- On-board refresh circuit that eliminates need for refresh signals on the LSI-11 bus.
- The system memory address space to which the option will respond is user-configured via switches contained on the module. An address can start at any 4K bank boundary.
- It will perform DATI, DATO, DATOB, DATIO, and DATIOB bus cycles according to LSI-11 bus protocol.
- Jumpers allow the module to be configured for using external bus refresh signals and disabling the internal refresh function.
- No special power is required. Only the normal +5 and +12 Vdc voltages are necessary. An on-board charge pump circuit provides the necessary −5 V operating voltage to the memory circuits.
- Jumpers allow the user to implement battery backup power.

SPECIFICATIONS

Identification	M7955-YD
Size	Quad

Power

System Power	Operating	Standby
+5 V ± 5%	1.1 A	1.1 A
+12 V ± 3%	0.54 A	0.16 A
Power	12 W	7.7 W

Battery Backup Power	
+5 V ± 5%	0.8 A
+12 V ± 3%	0.16 A

Bus Loads	
AC	2.3
DC	1

MSV11-CD

Operational

Operating Speed – The operating speeds stated below are based on the bus not attempting a memory cycle during a refresh operation and that an arbitration was not necessary.

Access Time

Bus Cycle Type	Access Time
DATI, DATIO	300 ns typ. (350 ns max. from RSYNC H)
DATO(B)	300 ns typ. (350 ns max. from RDOUT H).

Cycle Time

Bus Cycle Type	Cycle Time
DATI	650 ns typ. (750 ns max. from RSYNC H)
DATO(B)	650 ns typ. (750 ns max. from RDOUT H)

NOTE

If a bus cycle is being done as a result of winning the synchronization arbitration, the bus cycle will be delayed or increased from 0 to 80 ns. If a refresh operation is in progress when a cycle is requested, a delay from 0 to 750 ns will occur before the bus cycle starts.

CONFIGURATION

General

Configuring the MSV11-CD will alter its operation for a specific system application. The following items can be configured:

1. Select the starting address for the contiguous memory contained on the module
2. Select the number of banks required on the memory
3. Refresh mode: on-board or external refresh, and reply to external refresh signals
4. Battery backup power
5. Bus grant (BIAK L and BDMG L) signals.

In most applications, the user will simply configure the starting address and install the module. Refer to the following paragraphs for procedures for configuring each item.

MSV11-CD

Address Selection

The MSV11-CD address can start at any 4K bank boundary. The address configured is the starting address for the contiguous 16K portion of memory contained on the module. Set the switches, located as shown on Figure 1, to the desired starting address as listed in Table 1. The upper 4K address space is normally reserved for peripheral device and register addresses. Thus, bank 7 (addresses 160000–177777) normally should not be used for system memory in a 32K addressable system.

Number of Banks Selection

It may be necessary in some systems to use less than the 16K of memory that is available on the MSV11-CD. The 4K banks of memory can only be disabled consecutively from the top bank down. Table 2 identifies the proper configuration of the jumpers.

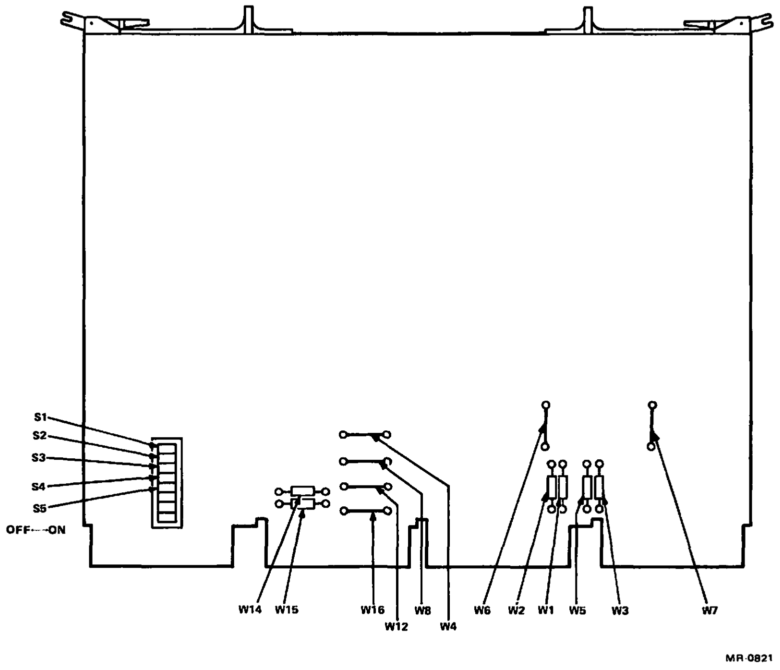


Figure MSV11-CD-1
MSV11-CD Switch and Jumper Locations

MSV11-CD

Table 1 MSV11-CD Addressing Summary

Starting Address	Banks	Address Range	Switch Settings S1	S2	S3	S4	S5
0	0-3	0-77777	1	1	1	1	1
20000	1-4	20000-117777	0	1	1	1	1
40000	2-5	40000-137777	1	0	1	1	1
60000	3-6	60000-157777	0	0	1	1	1
100000	4-7	100000-177777	1	1	0	1	1
120000	5-10	120000-217777	0	1	0	1	1
140000	6-11	140000-237777	1	0	0	1	1
160000	7-12	160000-257777	0	0	0	1	1
200000	10-13	200000-277777	1	1	1	0	1
220000	11-14	220000-317777	0	1	1	0	1
240000	12-15	240000-337777	1	0	1	0	1
260000	13-16	260000-357777	0	0	1	0	1
300000	14-17	300000-377777	1	1	0	0	1
320000	15-20	320000-417777	0	1	0	0	1
340000	16-21	340000-437777	1	0	0	0	1
360000	17-22	360000-457777	0	0	0	0	1
400000	20-23	400000-477777	1	1	1	1	0
420000	21-24	420000-517777	0	1	1	1	0
440000	22-25	440000-537777	1	0	1	1	0
460000	23-26	460000-557777	0	0	1	1	0
500000	24-27	500000-577777	1	1	0	1	0
520000	25-30	520000-617777	0	1	0	1	0
540000	26-31	540000-637777	1	0	0	1	0
560000	27-32	560000-657777	0	0	0	1	0
600000	30-33	600000-677777	1	1	1	0	0
620000	31-34	620000-717777	0	1	1	0	0
640000	32-35	640000-737777	1	0	1	0	0
660000	33-36	660000-757777	0	0	1	0	0
700000	34-37	700000-777777	1	1	0	0	0
720000	x	x-x	0	1	0	0	0
740000	x	x-x	1	0	0	0	0
760000	x	x-x	0	0	0	0	0

NOTES
1. Switch settings:
   1 = ON
   0 = OFF
2. Each memory bank = one 4K address space.

MSV11-CD

Table 2 Bank Selection

Banks Disabled	W4	W8	W12	W16
None	I	I	I	I
3	I	I	I	R
2, 3	I	I	R	R·
1, 2, 3	I	R	R	R

Refresh Mode Selection

The MSV11-CD module is factory-configured for internal (on-board) memory refresh and no reply (assertion of BRPLY L) in response to refresh cycles on the LSI-11 bus. Factory-installed wire-wrap jumpers W6 and W7, located as shown in Figure 1, provide these functions. W7, when installed, enables internal refresh. W6, when removed, enables the MSV11-CD to reply to external (LSI-11 bus) refresh cycles. W6 must be installed if W7 is installed to prevent erroneous assertion of the BRPLY L signal. A summary of refresh mode jumpers is provided in Table 3.

The reply to external refresh cycles is normally enabled when the module is deemed the slowest dynamic MOS memory device in the system. The slowest device (in this application) is generally the device located the greatest electrical distance from the device generating the refresh bus cycles. Only one device should be permitted to reply to refresh bus signals.

Table 3 Refresh Mode Selection

Jumper		
W6	W7	Refresh Mode Function
In	In	Factory configuration. Internal refresh; no reply.
Out	In	Illegal – do not use.
In	Out	External refresh; no reply.
Out	Out	External refresh; reply enabled.

Battery Backup

The MSV11-CD module is designed so that the dc power required to support it during a backup period is minimized. Jumper wires are provided on this module to allow the user to disconnect the memory and refresh logic from the normal bus power and connect it to a separate battery backup system. When supplied by DIGITAL, these jumpers allow the memory and refresh logic to be powered off the normal bus backplane power by having all power jumpers (W1, W2, W3, and W5) installed. The jumpers connect normal system power to battery backup power pins. If battery backup is to be used with the MSV11-CD, cut or remove jumpers W1 and W5 as described below. The module will then receive dc operating voltages (+5 V and +12 V) from the battery backup source.

To support battery backup, the user-configurable jumpers shown in Figure 1 should be configured as follows.

W1, W5	Remove to separate the battery power from the bused system power.
W2, W3	Insert to connect the battery power to the refresh logic.
W6, W7	Insert to enable internal refresh and to prevent the module from asserting BRPLY during refresh.

If battery backup power is available but not desired for a particular MSV11-CD module, cut or remove jumpers W2 and W3; jumpers W1 and W5 must remain installed.

To use the MSV11-CD in a battery backup system, the battery system must be capable of supplying the following power.

	1 Module	**2 Modules**
+5 Vdc ± 5%	0.8 A	1.6 A
+12 Vdc ± 3%	0.18 A	0.32 A

These voltages must remain within ±3 percent of the voltages for the LSI-11 bus at all times and must not change by more than ±3 percent during the transition to or from the battery. One MSV11-CD module draws approximately 7.5 W of power while in the backup mode. This means that for a typical backup system that is 30 percent efficient, a 2.5 A–hr battery will support one module for approximately 1.5 hours.

When used in a PDP-11V03 system or equivalent, no additional cooling of this module is required during the backup period if the room temperature is maintained at less than 36° C (97° F) for one module and at less than 28° C (82° F) for two modules.

Bus Grant Continuity

Bus grant continuity jumpers W14 and W15 are factory-installed and normally should not be removed.

MSV11-CD

FUNCTIONAL DESCRIPTION

General
The basic functions that comprise the MSV11-CD are shown in Figure 2. All signals interface with the LSI-11 bus via bus receivers, bus drivers, and bus transceivers, which contain both receiver and driver functions. The receiver function of the bus transceivers shown in the figure distributes the 16 input data/address lines (DI00–15 H) to the memory array and to addressing logic.

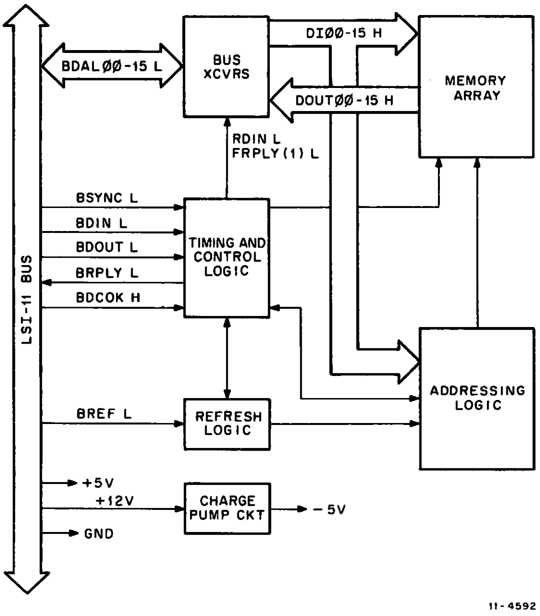


Figure MSV11-CD-2
MSV11-CD Block Diagram

Timing and control logic receives BSEL H from the addressing logic whenever the bus address received is within the range configured by the operator. This is the signal that allows the timing and control logic to communicate with the bus and generate appropriate timing and control signals from the option.

The memory responds to address and control signals by storing (writing) an 8-bit byte or 16-bit word, or by outputting 16-bit (word) read data to the LSI-11 bus.

Additional functions include refresh logic and a charge pump circuit. The refresh logic is capable of responding to LSI-11 bus ("external") refresh signals, or it can produce refresh signals (including row address) independent of the LSI-11 bus ("internal refresh"). Memory chips are refreshed by forcing a read cycle at all 64 row addresses once every 2 ms, maximum. The MSV11-CD refresh logic refreshes one row address every 25 μs.

The charge pump circuit eliminates the need for backplane power other than the standard +5 V and +12 V. It contains an inverter circuit that receives +12 V input power and produces a negative voltage output. A zener-diode regulator produces −5 V for the memory integrated circuits.

Memory Array

The memory array (Figure 3) consists of sixteen 4K by 1-bit integrated circuits per memory bank; four memory banks are included in the option. Hence, 64 integrated circuits comprise the array. Each integrated circuit provides 1-bit by 4096-location storage.

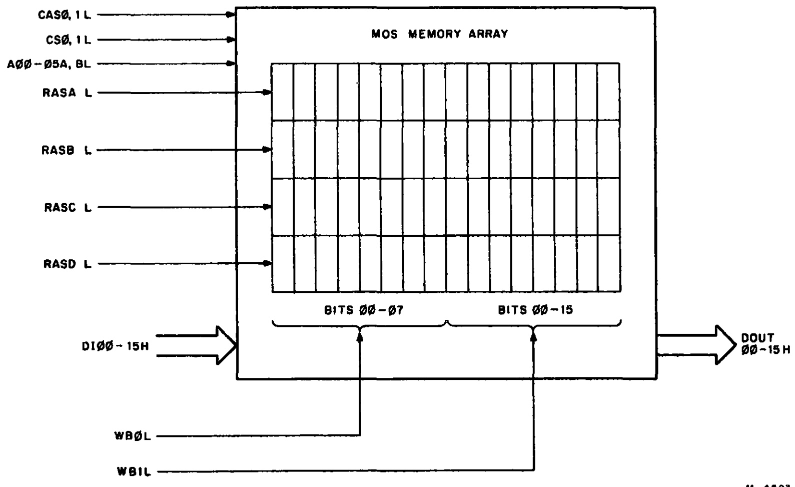


Figure MSV11-CD-3
Memory Array

The memory array is arranged so that one bank out of four can be accessed at a time. The row address strobe (RASAL, RASBL, RASCL, and RASDL) signals select the desired bank. During a write operation, control signals WB0 L and WB1 L enable writing in the "low" byte (data bits 00–07) or "high" byte (data bits 08–15), or full word (both bytes accessed simultaneously).

Duplicate drivers are used for certain signals. These signals include column address strobe (CAS), chip select (CS), and address lines 00–05.

The duplicate driver output signals include:

Drivers	Output Signals
CAS	CAS0 L and CAS1 L
CS	CS0 L and CS1 L
Address	A00–05A L and A00–05B L

Each memory integrated circuit has six address input pins and contains a 12-bit address latch. Addressing the 4096 locations is accomplished by multiplexing the 12-bit address in two 6-bit segments. The low-order six bits (bus address bits 01–06) are first multiplexed onto A00–05A L/A00–05B L signal lines and latched in the addressed 16 integrated circuits by the active row address strobe signal. This is followed by multiplexing the high-order six bus address bits 07–12 onto the same six address lines; the column address strobe signals latch the address bits in the memory integrated circuits. The memory array is ready for the read or write operation once the addressing sequence has been completed. A detailed description of the complete addressing and read or write sequence is included in the following paragraphs.

Addressing Logic
The addressing logic (Figure 4) decodes a memory bank within the user-defined address space, latches the word or byte address within the selected bank, and routes time-multiplexed 6-bit address segments to the memory array for 1 of 4096 word addresses within the bank. In addition, it routes the refresh address to the memory bank during a refresh cycle. The following paragraphs describe addressing logic functions in detail. Generation of the control signals and how they relate to a complete memory read or write operation are described in the paragraph entitled "Timing and Control Logic."

A memory read or write cycle is initiated by the addressing portion of the LSI-11 bus cycle. The "bus master" first places the address on BDAL00–15 L and BAD16 and 17 L. (BAD16 and 17 L are presently not used by LSI-11 system hardware, but are available for future system configurations.) The bus master then asserts BSYNC L, indicating that a valid address is on the bus. LATCH L occurs on the leading edge of

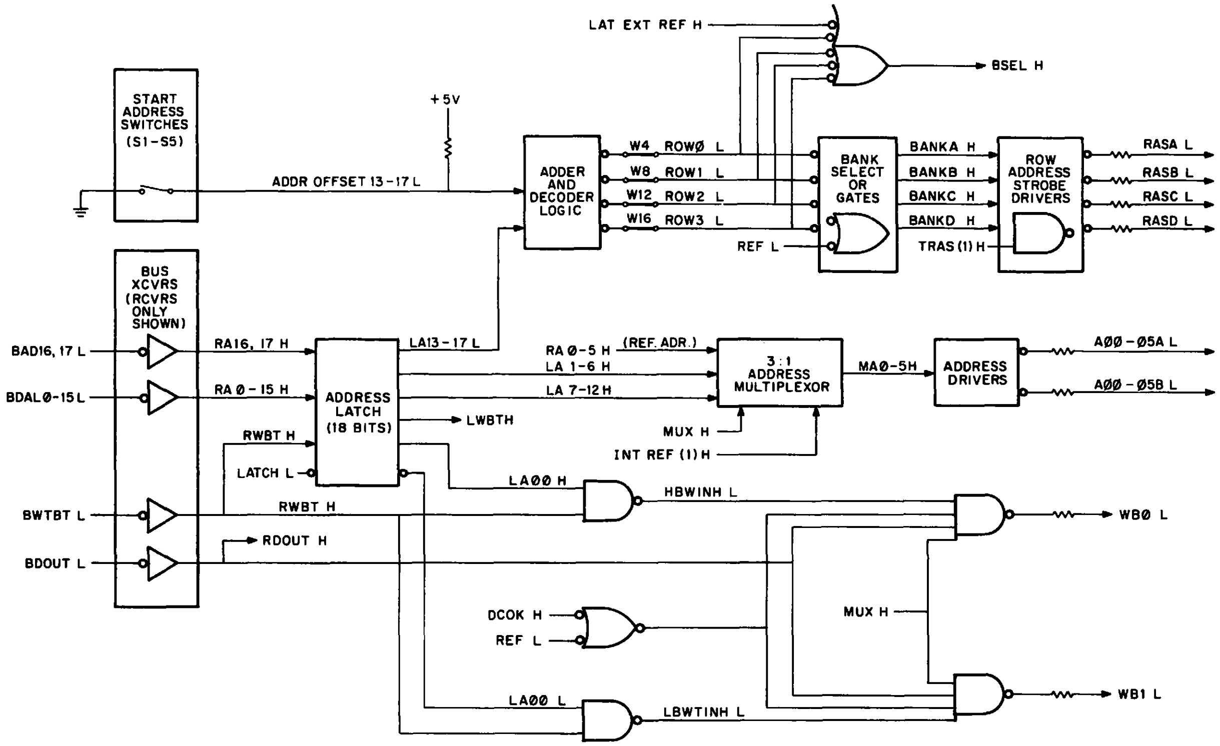


Figure MSV11-CD-4
MSV11-CD Addressing Logic

BSYNC L, latching the received address bits RA0–17 H and the received bus write/byte control signal, RWBT H, for the duration of the bus cycle. Note that RWBT H is active during the addressing portion of DATO or DATOB bus cycles, indicating that a memory write operation will follow.

Start address switches S1–S5 are configured by the user to select the address space that the 16K memory will occupy. ADDR OFFSET 13–17 L signals are the encoded starting address switch bits. The bits are applied to the adder and decoder logic, where they are added to latched address bits LA13–17 L. One decoded output (ROW0–3 L) goes active (low) only when the latched address is within the 16K address space selected by the user. When the latched address is not within the assigned address space, the decoder outputs remain passive (high). These signals are applied to the BSEL H and bank select OR gates. Any active output (or an active LAT EXT REF L signal) produces an active (high) BSEL H signal; BSEL H enables the timing and control logic to start a memory cycle.

Bank select OR gates receive one active ROW signal via jumper W4, W8, W12, or W16 or an active REF L signal and produce one active BANK A-D H for normal memory cycles, or all active BANK A-D H signals during a memory refresh operation. The active BANK A-D H signal is ANDed with TRAS (1) H for proper timing during the memory cycle. The appropriate row address driver then produces one active RAS A-D L signal that enables one memory bank; all four signals go to the active state during a refresh operation to allow all memory banks to be refreshed simultaneously.

The 3:1 address multiplexer and address drivers select and apply the 6-bit address to the memory array. During a normal (non-refresh) memory cycle, MUX H and INT REF (1) H are passive (low), and the low-order six address bits are applied to the memory array and latched in each memory integrated circuit. MUX H then goes high, selecting the high-order six bits, which are then latched in the memory integrated circuits to comprise the 12-bit address. During an internal memory refresh operation, INT REF (1) H goes active, selecting the refresh address bits (RA0–5 H), which are applied to the memory array. If the external memory refresh mode of operation is configured by the user, the low-order address bits (LA1–6 H) are used for the refresh address.

During a write operation, the bus I/O sequence can be a DATO (word) or DATOB (byte) cycle. Bus signal BWTBT L goes passive during the data portion of a DATO cycle, enabling writing in all 16 bits of the addressed word. RWBT H goes low, inhibiting the byte inhibit gates, and HBWINH L and LBWINH L go passive (high), enabling WB0 L and WB1 L memory array drivers. The drivers are enabled only during an output data transfer (BDOUT L is active) and a non-refresh operation. The active MUX H signal gates the drivers on at the proper time during the bus cycle. During a DATOB bus cycle, RWBT H goes active, enabling the byte inhibit gates. Latched address bit 0 (signals LA00 H and LA00 L) inhibits the WB0 L or WB1 L signals, as appropriate, to enable writing in only the addressed 8-bit byte within the addressed 16-bit location.

When backplane power is abnormal, DCOK H goes low. This signal inhibits WB0 L and WB1 L, preventing erroneous write operations.

Timing and Control Logic

Timing and control logic (Figure 5) interfaces the MSV11-CD to the LSI-11 bus and produces the internal control signals required for memory operation. This function produces timing and control signals for three types of memory cycles: read (DATI), write (DATO or DATOB), and refresh. In addition, read-modify-write cycles (DATIO or DATIOB) can be executed according to LSI-11 bus protocol. The MSV11-CD will also respond to externally generated refresh cycles if that refresh mode is configured by the user.

The control signal sequence for a memory read operation is shown in Figure 6. All bus transactions involving the MSV11-CD are initiated by the addressing portion of the bus cycle. The bus master first places an address on the BDAL00–15 L (and optional BAD16, 17 L) lines and asserts BSYNC L. BSYNC L is received and distributed as RSYNC H and LATCH L. The leading edge of LATCH L latches the address in the addressing logic and the address will remain valid for the duration of the bus transaction (during the active state of BSYNC L).

Initially, NO DIN L and LOCKOUT L are passive (high), enabling two inputs of the read cycle initiate gate. RSYNC H enables a third input to the gate. When the latched address is within the MSV11-CD's address space configured by the user, BSEL H goes high, producing a low (active) read cycle initiate gate output. The low signal is ORed, producing an active GOTIM H signal. This is the signal that starts the memory cycle.

GOTIM H is then applied to the 80-ns delay circuit. The delay circuit inserts an 80-ns delay only during internal refresh operation. The delay circuit is shown in Figure 7.

If an internal refresh cycle is in progress when BSYNC L is asserted, the memory read or write cycle cannot be executed until the refresh cycle has been completed. However, if an internal refresh cycle is requested after GOTIM H is generated for a read or write cycle, the read or write cycle is first executed.

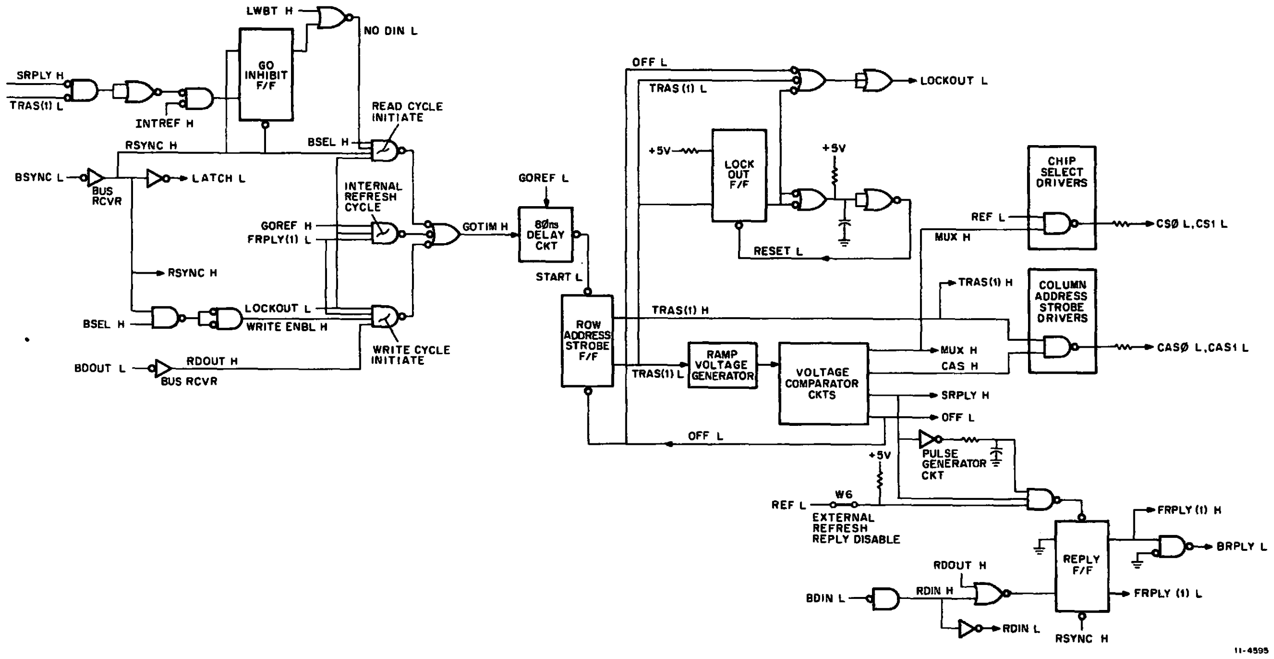


Figure MSV11-CD-5
MSV11-CD Timing and Control Logic

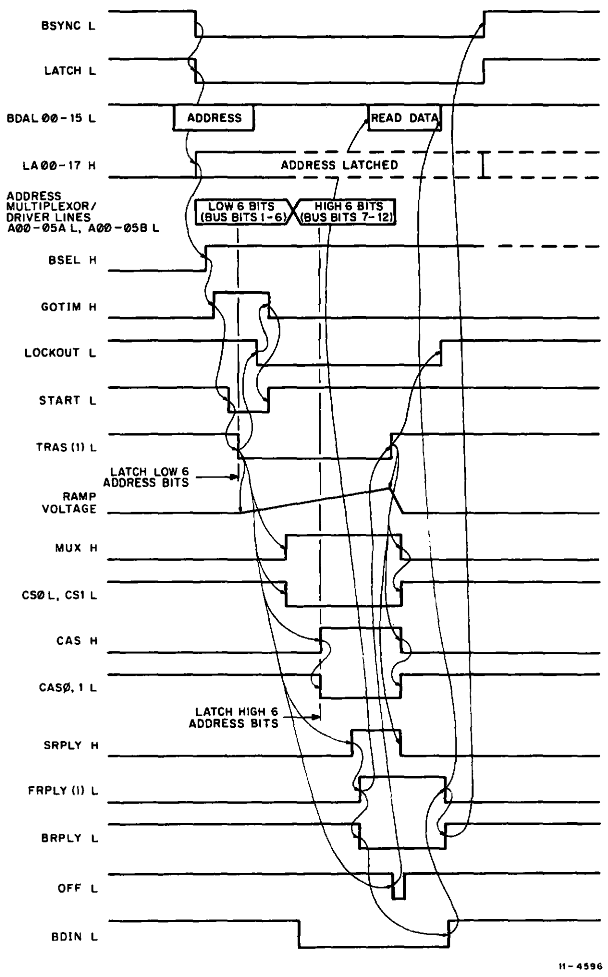


Figure MSV11-CD-6
MSV11-CD Read (DATI) Sequence

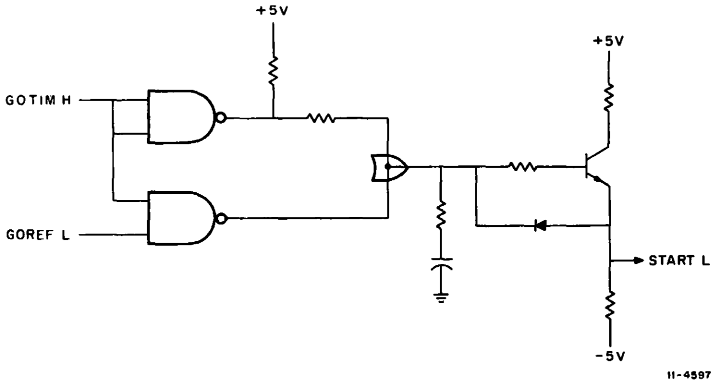


Figure MSV11-CD-7
80 ns Delay Circuit

START L is the active (low) delay circuit output and it sets the row address strobe flip-flop. Active TRAS (1) H and TRAS (1) L signals are produced. Active TRAS (1) L is ORed with OFF L and the lockout flip-flop to generate LOCKOUT L. LOCKOUT L is ANDed in the read cycle initiate gate, the write cycle initiate gate, and the internal refresh gate to inhibit the GOTIM H signal from occurring until the present cycle is complete. The lockout flip-flop is used to generate a delay from the reset of TRAS (1) L to ensure proper memory timing. TRAS (1) L is gated with passive (low) SRPLY H and INTREF H signals, producing a high signal that clocks the Go Inhibit flip-flop to the set state; note that the high (active) RSYNC H signal applied to the data input of the flip-flop when clocked by the leading edge of TRAS (1) L enables the set state on the positive-going leading edge of the clock input to the flip-flop. The resulting high flip-flop output signal is ORed to produce an active (low) NO DIN L signal that inhibits the read cycle initiate gate to prevent multiple DATI cycles from occurring without RSYNC H being reset.

TRAS (1) H enables column address strobe drivers that latch the high-order 6-bit address in the memory array later in the read cycle.

TRAS (1) L activates an R-C ramp voltage generator that applies a rising voltage (Figure 6) to the voltage comparator circuits. Four voltage comparators, each having different reference voltage inputs, produce the four control signals: MUX H, CAS H, SRPLY H, and OFF L. The reference voltage applied to a comparator determines the point along the ramp voltage at which the comparator's output will change logical state.

Hence, the reference voltage, relative to the ramp voltage, determines the time delay produced by each comparator circuit.

The first active signal produced by the comparator circuit is MUX H. MUX H selects the high-order six address bits that are applied to the memory array. MUX H is ANDed with the passive REF L signal, producing the active chip select (CS0 L and CS1 L) signals.

The next voltage comparator output signal that goes active is CAS H. CAS H is ANDed with the active TRAS (1) H signal, producing the active column address strobe (CAS0 L and CAS1 L) signals. These signals latch the high-order six address bits in the memory array integrated circuits, completing the 12-bit address within the accessed 4K bank. Read data is then available on DOUT00–15 H.

The next voltage comparator output signal that goes active is SRPLY H. SRPLY H is applied to a pulse generator circuit whose 30 ns output pulse is ANDed with the passive REF L signal; the resulting low pulse sets the reply flip-flop, and FRPLY (1) H and FRPLY (1) L signals go to their active states. The BRPLY L bus driver asserts the BRPLY L signal and driver portions of the bus transceivers, enabled by RDIN L (received BDIN L) and FRPLY (1) L, place the memory read data on BDAL00–15 L.

The last voltage comparator signal produced is OFF L. OFF L clears the row address strobe flip-flop and TRAS (1) H and TRAS (1) L go to their passive states. The passive (low) TRAS (1) H signal inhibits the column address strobe drivers and CAS0 L and CAS1 L go to their passive states. The passive (high) TRAS (1) L signal resets the ramp voltage and MUX H, CS0 L, CS1 L, CAS H, SRPLY H, and OFF L go to their passive states.

The bus master responds to the active BRPLY L signal by reading the data from the bus and terminating the BDIN L signal. BDIN L (passive) is received and inverted, producing a negative-going trailing edge on BDIN H. BDIN H is ORed with RDOUT H, producing a positive-going transition at the clock input of the reply flip-flop. This transition clocks the flip-flop to the reset state. FRPLY (1) H goes low, inhibiting the BRPLY L bus driver and FRPLY (1) L goes high, inhibiting the BDAL00–15 L read data bus drivers. The bus master responds by terminating BSYNC L, and the memory read cycle is completed.

A memory write (DATO or DATOB) cycle is somewhat similar to the memory read cycle. However, during the addressing portion of the cycle, the bus master asserts BWTBT L, indicating an output (write) operation is to follow. The signal sequence for the timing and control logic write operation is shown in Figure 8. BWTBT is received (RWBT H) and

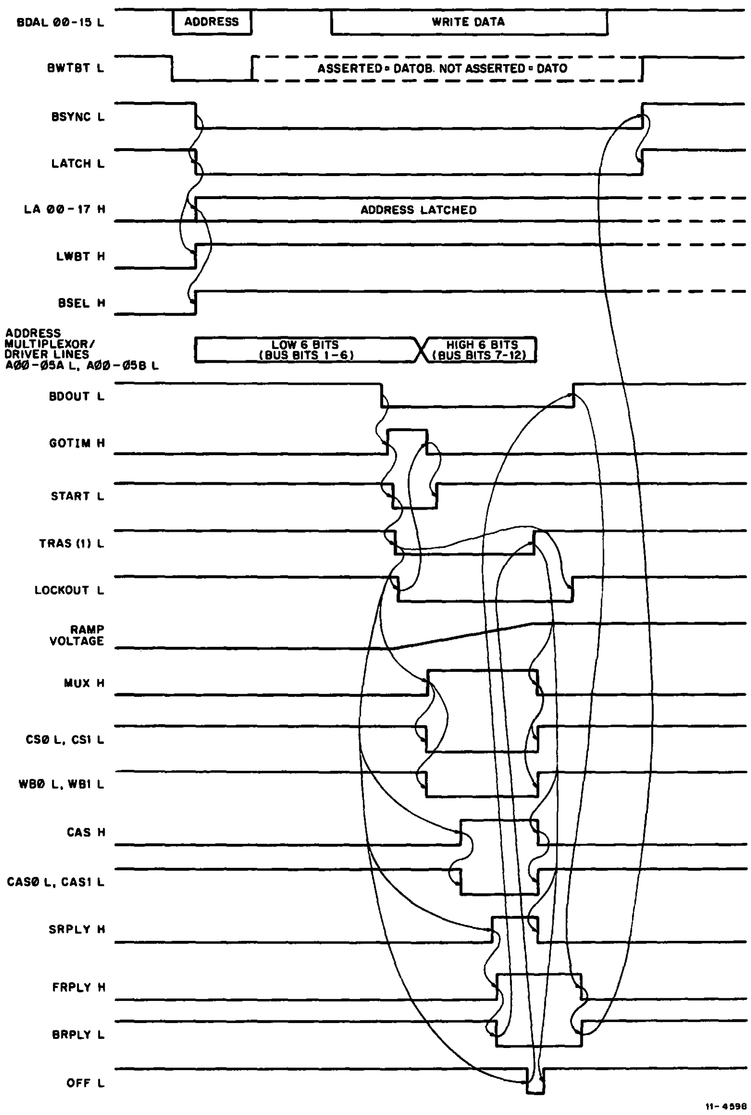


Figure MSV11-CD-8
MSV11-CD Write (DATO or DATOB) Sequence

latched by the active LATCH L signal; the address bits are latched in the same manner as previously described. The latched signal (LWBT H) inhibits the read cycle initiate gate and prevents immediate generation of the GOTIM H signal. However, the latched row address bits LA1–6 H are routed via the 3:1 address multiplexer to the memory array.

The bus master then places the write data on the BDAL bus and asserts BDOUT L. BDOUT L is received, producing RDOUT H. RDOUT H is gated with passive (high) LOCKOUT L, FRPLY (1) L, and active WRITE ENBL H signals to produce the active GOTIM H signal.

Operation of the timing and control logic continues as described for read operation generation of the MUX H, CAS H, SRPLY H, and OFF L signals. However, the RDOUT signal enables WBO L and/or WB1 L signal generation in the addressing logic, and the received data is written in the addressed location.

The bus master responds to the BRPLY L signal by terminating BDOUT L and removing the write data from the BDAL bus lines. RDOUT H, which is ORed with the passive RDIN H signal, goes low, and the low to high transition resets the reply flip-flop. BRPLY L then goes passive (high). The bus master responds by terminating BSYNC L and the write operation is completed.

The memory read-modify-write operation is produced by the DATIO (or DATIOB) bus cycle. The bus master first initiates a "memory read" operation at the addressed location. However, instead of terminating BSYNC L after the read portion of the cycle has been completed, BSYNC L remains asserted; the bus master places the write (modified) data on the bus and asserts BDOUT L. The MSV11-CD responds by writing the data in the addressed word or byte location. The BWTBT L signal is asserted with the write data to indicate a DATIO bus cycle is in progress, when appropriate. Except for the addressing portion being omitted for the write portion of the DATIO (or DATIOB) bus cycle, operation of the MSV11-CD is as previously described for memory read and memory write operations.

Memory Refresh Operation

Dynamic MOS memory integrated circuits comprising the memory array require periodic refreshing in order to retain stored data. This is accomplished by forcing a memory read operation on each of the 64 row addresses; one read operation is required for each row address. Each row address must be refreshed in this manner once every 2 ms (maximum).

MSV11-CD memory refresh can be accomplished by using the on-board (internal) refresh logic signals, or by using refresh signals present on the

LSI-11 bus (external refresh). The internal refresh circuit is most transparent to the LSI-11 system since it automatically refreshes one row address every 25 μs; bus-generated refresh cycles are not required when the internal refresh mode is selected. When other memory system components in the system require LSI-11 bus memory refresh signals, the MSV11-CD internal refresh mode can be disabled and the refresh operation can be controlled externally by LSI-11 bus signals. Note thatwhen the external refresh mode is selected, all system memory components are refreshed simultaneously.

The MSV11-CD refresh logic circuit is shown in Figure 9. The sequence of internal refresh operation is shown in Figure 10. When the internal refresh mode is selected, jumper W7 is installed, producing the active (high) CLK EN H signal. This signal activates the 25 μs clock and places a high level at the internal refresh cycle flip-flop's data input. The positive-going transition of the 25 μs clock clocks the flip-flop to the set state, causing GO REF L and GO REF H to go to their active states (low and high, respectively). GO REF H conditions the data input of the internal refresh flip-flop.

When the MSV11-CD is not involved in a read or write operation, FRPLY (1) L and LOCKOUT L are passive (Figure 5). These passive signals are gated with the active GO REF H signal, producing an active GOTIM H signal. The leading edge of GOTIM H clocks the internal refresh flip-flop to the set state and INTREF (1) H goes high. INTREF (1) H is ORed with the passive LAT EXT REF H signal to produce an active (low) REF L signal.

GO REF L is applied to the 80 ns delay circuit (Figure 7), forcing the 80 ns delay to occur. The delay provides settling time for the internal refresh flip-flop. After the 80 ns delay has completed, START L goes active, setting the row address strobe flip-flop, and TRAS (1) L and TRAS (1) H go to their active states. TRAS (1) L provides an active LOCKOUT L signal, inhibiting the GOTIM H signal. Operation then continues as in the memory read cycle, except the active INTREF (1) H signal selects RA00–05 H refresh address (row) bits, which are routed through the address multiplexer and applied to the memory array; passive (low) column address bits are applied to the memory array during the active state of MUX H, but are not significant during the refresh operation. The active REF L signal, applied to the reply circuit via jumper W6, also inhibits SRPLY H from setting the reply flip-flop and generating an erroneous BRPLY L signal.

SRPLY H is gated with INTREF (1) H, producing a low clear signal for the internal refresh cycle flip-flop, and GO REF L and GO REF H go passive. GOTIM H goes passive and the 80 ns delay circuit START L signal goes passive. OFF L then goes active (low), clearing the internal

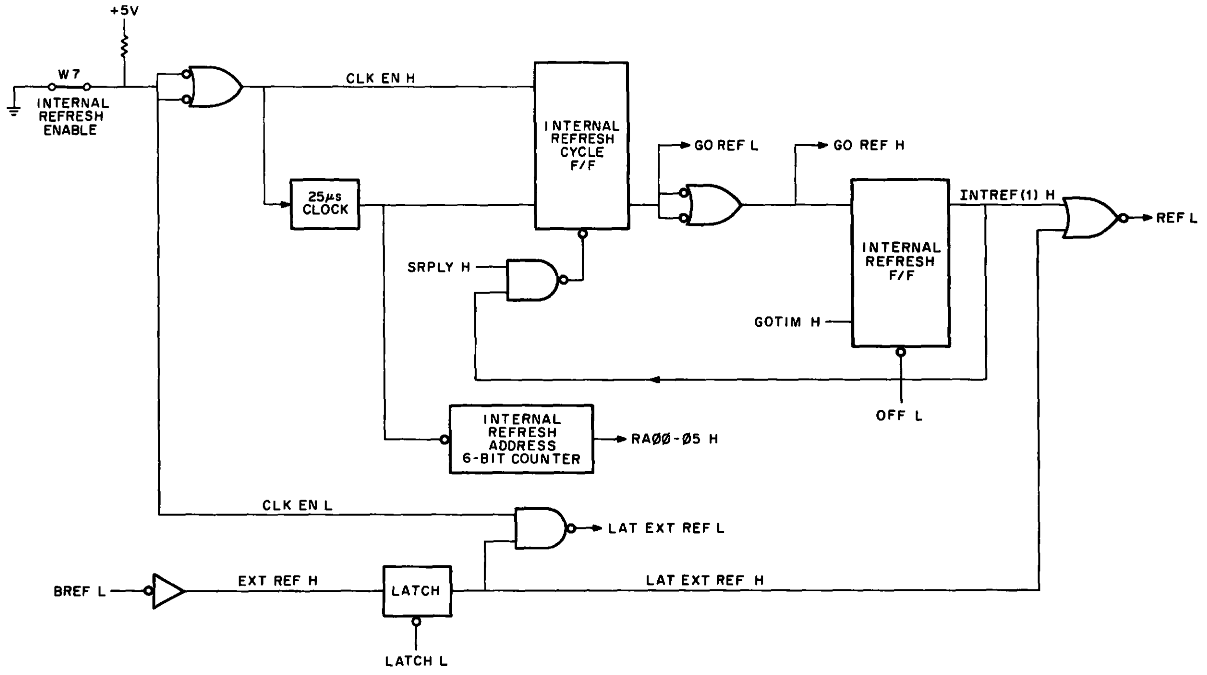
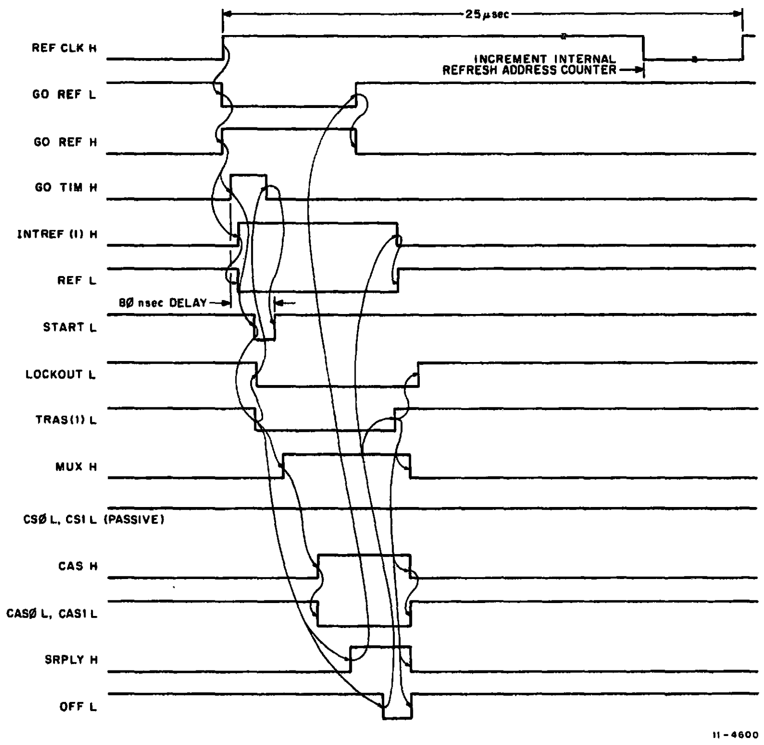


Figure MSV11-CD-9
MSV11-CD Refresh Logic



MSV11-CD Internal Refresh Sequence

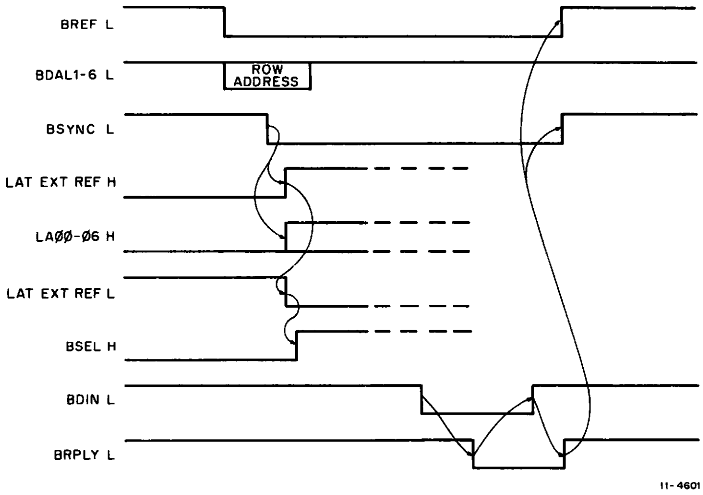
refresh and row address strobe flip-flops; REF L, TRAS (1) L, LOCKOUT L, MUX H, CAS H, SRPLY H, and OFF L signals go to their passive states.

The internal refresh counter is incremented by the trailing edge of REF CLK H prior to the next internal refresh operation. Hence, each successive internal refresh operation uses the next sequential row address.

When a memory read or write operation is initiated just prior to the refresh operation, and the read or write cycle is still in progress, the leading edge of GOTIM H clocks the passive (low) GO REF H signal into the internal refresh flip-flop; INTREF (1) H remains reset for the duration of the memory cycle. The refresh operation will not be enabled until LOCKOUT L and FRPLY (1) L go passive (high) at the end of the read or write cycle, enabling a new GOTIM H signal. If this condition occurs, the leading edge of the new GOTIM H signal clocks the active (high) GO REF F signal into the internal refresh flip-flop, and the refresh operation continues as described previously.

When the external refresh mode is selected, jumper W7 is removed. CLK EN H goes low, and the internal refresh cycle flip-flop cannot be set by the 25 μs clock signal. CLK EN L goes high, enabling one input to the LAT EXT REF L gate.

An external refresh cycle is initiated by the bus master asserting BREF L; EXT REF H goes high. The bus master places the refresh row address on the BDAL 1-6 L lines and asserts BSYNC L. BSYNC L produces the active (low) LATCH L signal that latches the 6-bit address in the address latch (Figure 4), and the EXT REF H signal (Figure 9), producing the active (high) LAT EXT REF H signal. This signal is gated with CLK EN L, producing the LAT EXT REF L signal and forcing an active BSEL H signal in the addressing logic. Thus, the MSV11-CD is addressed and enabled for the refresh operation. The external refresh signal sequence is shown in Figure 11.

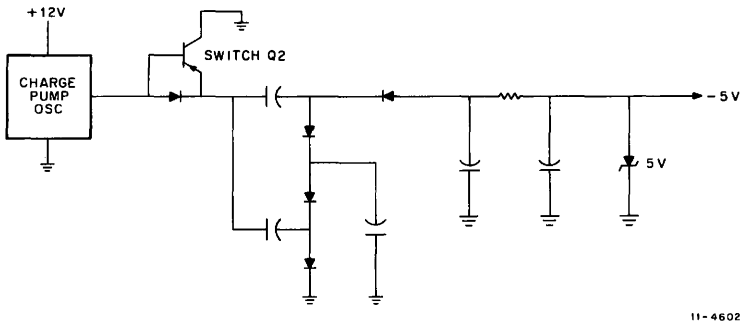


MSV11-CD External Refresh Signal Sequence

The refresh operation continues as described for a memory read operation, except the active REF L signal (produced by the active LAT EXT REF H signals) inhibits a BRPLY L signal, as described for the internal refresh operation. If the MSV11-CD must reply during the external refresh bus transaction, jumper W6 is removed; REF L will not inhibit the reply circuit and BRPLY L will be generated. Only one MOS memory module in a system is required to reply to the refresh bus transactions. The module that should reply is the module with the slowest access time relative to the bus master device. This module is generally the module located the greatest electrical distance on the LSI-11 bus from the bus master device controlling the refresh operation.

Charge Pump Circuit

The charge pump circuit (Figure 12) provides the –5 V power for the MOS memory array integrated circuits. The input power source for this circuit is +12 V. An oscillator circuit, running at approximately 40 kHz, produces a square-wave drive signal to the rectifier circuit. Q2 switches the ground alternation for the charge pump capacitors, reducing the switching current requirement for the oscillator circuit. The rectifier output is –10 V (approximately). The –5 V output is zener-diode regulated. Note that this circuit eliminates the need for backplane power other than the standard +5 V and +12 V. The circuit remains active when battery backup power is used.



MSV11-CD Charge Pump Circuit

MSV11-D, -E MEMORY

GENERAL

All MSV11-D and MSV11-E memory modules plug into any LSI-11 bus. There are eight versions of this module and they are listed below.

Model	Memory Capacity	Module	Parity Bits
MSV11-DA	4K by 16 bits	M8044-A	No
MSV11-DB	8K by 16 bits	M8044-B	No
MSV11-DC	16K by 16 bits	M8044-C	No
MSV11-DD	32K by 16 bits	M8044-D	No
MSV11-EA	4K by 18 bits	M8045-A	Yes
MSV11-EB	8K by 18 bits	M8045-B	Yes
MSV11-EC	16K by 18 bits	M8045-C	Yes
MSV11-ED	32K by 18 bits	M8045-D	Yes

NOTE
K = 1024 (e.g., 4K = 4096).

Memory storage is provided by either 4K by 1 bit or 16K by 1 bit integrated circuits, depending on model. The integrated circuits are dynamic metal oxide semiconductor (MOS) types that the processor can access during read and write operations. Memory contents are volatile; that is, when operating power is lost, memory data is lost. However, memory contents can be protected during system power failures by supplying battery backup power.

FEATURES

- On-board memory refresh is provided, eliminating the need for refresh signals on the LSI-11 bus.

- The system memory address space to which the module will respond is user-configured via switches contained on the module. An address can start at any 4K bank boundary ranging through the 0–128K address range.

- Memory modules perform DATI, DATO, DATOB, DATIO, and DATIOB bus cycles according to LSI-11 bus protocol.

- No special power is required. Only the normal +5 V and +12 V present on the LSI-11 backplane are necessary. An on-board charge pump circuit produces the necessary –5 V operating voltage for the memory integrated circuits.

MSV11-D, -E

- Jumpers allow the user to implement battery backup power.
- Memory access is disabled when "bank 7" is addressed (BBS7 L is asserted) or when "external" memory refresh bus cycles are in progress. The lower 2K of bank 7 can be enabled by a user-installed jumper. (Bank 7, the upper 4K system address space, is normally reserved for peripheral device addressing.)

SPECIFICATIONS

Identification	
MSV11-DX	M8044-X
MSV11-EX	M8045-X (X = A,B,C,D)
Size	Double

Power

	Supply Voltage	Typical Operating Power	Typical Standby Power
MSV11-DA,-DC (4K or 16K)	+5 V system power	1.7 A	1.7 A
	+5 V battery backup	0.7 A	0.7 A
	+12 V system power or battery backup	0.34 A	0.06 A
MSV11-DB,-DD (8K or 32K)	+5 V system power	1.7 A	1.7 A
	+5 V battery backup	0.7 A	0.7 A
	+12 V system power or battery backup	0.37 A	0.08 A
MSV11-EA,-EB (4K or 16K)	+5 V system power	2.0 A	2.0 A
	+5 V battery backup	1.0 A	1.0 A
	+12 V system power or battery backup	0.38 A	0.06 A
MSV11-EC,-ED (8K or 32K)	+5 V system power	2.0 A	2.0 A
	+5 V battery backup	1.0 A	1.0 A
	+12 V system power or battery backup	0.41 A	0.09 A

Bus Loading	
AC	2
DC	1

MSV11-D, -E

Operational

MSV11-D

Bus Cycle	**Access Time (ns)**			**Cycle Time (ns)**		
Type	**Typ**	**Max**	**Notes**	**Typ**	**Max**	**Notes**
DATI	210	225	1,2	500	520	1,4
DATO(B)	100	110	1,2	545	565	1,5
DATIO(B)	630	650	1,3	1075	1100	1,6

MSV11-E

Bus Cycle	**Access Time (ns)**			**Cycle Time (ns)**		
Type	**Typ**	**Max**	**Notes**	**Typ**	**Max**	**Notes**
DATI	250	265	1,2	500	520	1,4
DATO(B)	100	110	1,2	545	565	1,5
DATIO(B)	670	690	1,3	1115	1140	1,6

All Models

Refresh cycle time (7) 575 ns typ., 600 ns max.

NOTES

1. All operating speeds are in nanoseconds and are based on memory not busy and no refresh arbitration. Refresh arbitration adds 100 ns typical (120 ns maximum) to access and cycle times. Refresh conflicts add 575 ns typical (600 ns maximum) to access and cycle times.

2. Access times are defined as internal SYNC H to REPLY H with minimum times (25 or 50 ns) from SYNC H to DIN H or DOUT H. The DATO(B) access and cycle times assume a minimum 50 ns from SYNC H to DOUT H at bus receiver outputs. For actual LSI-11 bus measurements, 150 ns should be added to DATO(B) times, i.e., access time (typical) = 100 + 150 = 250 ns.

3. Access times are defined as internal SYNC H to REPLY H [DATO(B)] with minimum time (25 ns) from SYNC H to DIN H, and minimum time (350 ns) from RPLY H (DATI) asserted to DOUT H asserted.

4. Cycle times are defined as internal SYNC H to LOCKOUT L negated.

5. Cycle times are defined as internal SYNC H to LOCKOUT L negated with minimum time (50 ns) from SYNC H to DOUT H.

6. Cycle times are defined as internal SYNC H to LOCKOUT L [DATO(B)] with minimum times (25 ns) from SYNC H to DIN H and minimum time (350 ns) from RPLY H (DATI) asserted to DOUT asserted.

7. Refresh cycle time is defined as internal REF REQ L to LOCKOUT L negated.

CONFIGURATION

General

Configuring the MSV11-D or MSV11-E will alter its operation for a specific system application. The following items can be configured.

1. Select the starting address for the contiguous memory contained on the module

2. Battery backup power

3. Enable/disable 2K word portion of bank 7

NOTES

1. If the MSV11-D or MSV11-E memory module is installed in a system that contains a KD11-F processor module etch revision C or D, CS revision H2 or earlier, BDAL16 L and BDAL17 L bus lines (AC1 and AD1, respectively) must be terminated. An REV11-A, TEV11, or BCV1B cable set will also accomplish this termination.

2. Each MSV11-D or -E module contains two factory-installed wire-wrap jumpers that select memory size (4K, 8K, 16K, or 32K); these jumper configurations normally should not be changed.

MSV11-D, -E

Address Selection

The MSV11-D or MSV11-E address can start at any 4K bank boundary. The address configured is the starting address for the contiguous portion of memory (4K, 8K, 16K, or 32K) contained on the module. Set the switches, located as shown in Figure 1, to the desired starting address as listed in Table 1. The upper 4K address space is normally reserved for peripheral device and register addresses.

Factory-configured modules will not respond to bank 7 addresses. In special applications that permit the use of the lower 2K portion of bank 7 for system memory, enable the lower 2K portion of bank 7 by removing the jumper from wire-wrap pins 1 and 3 and connecting a new jumper from 1 to 2.

Battery Backup Power

The MSV11-D and MSV11-E modules are factory-configured with the power jumpers installed for normal system power. In this configuration, the memory and refresh logic are powered from the normal bus backplane power. The modules are designed so that the dc power required to support them during a backup period is minimized. The jumpers are provided to allow the user to disconnect the memory and refresh logic from the normal bus power and connect it to a separate battery source. The user can configure the jumpers, shown in Figure 1, for battery backup operation as follows:

W2, W3	Remove to separate the module from the bus-powered backplane.
W4, W5	Insert to connect the battery power to the memory and refresh logic.

To use the MSV11-D or MSV11-E modules in a battery backup system, the battery or source must be capable of supplying the following:

	MSV11-D (1 Module)	**MSV11-E (1 Module)**
+5 Vdc ± 3%	0.7 A	1.0 A
+12 Vdc ± 5%	0.37 A	0.41 A max

These voltages must remain within ±3% of the bus voltage at all times and not vary more than ±3% during the transition to or from the battery.

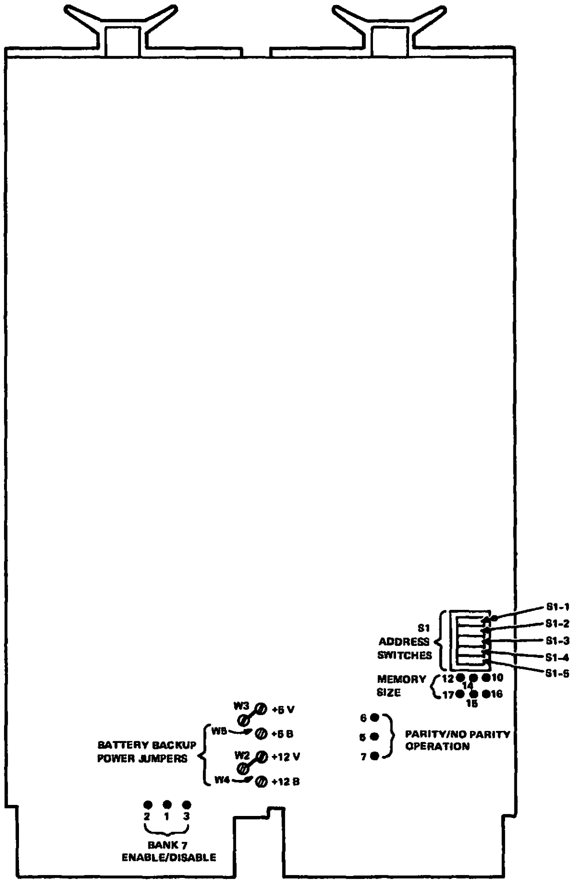


Figure MSV11-D, -E-1
MSV11-D, MSV11-E Switch and Jumper Locations

Table 1 MSV11-D, MSV11-E Addressing Summary

Starting	Switch Settings					Memory Bank(s) Selected			
Address	S1-1	S1-2	S1-3	S1-4	S1-5	-DA, -EA	-DB, -EB	-DC, -EC	-DD, -ED
0	C	C	C	C	C	0	0–1	0–3	0–7
20000	C	C	C	C	O	1	1-2	1-4	1-10
40000	C	C	C	O	C	2	2-3	2-5	2-11
60000	C	C	C	O	O	3	3-4	3-6	3-12
100000	C	C	O	C	C	4	4-5	4-7	4-13
120000	C	C	O	C	O	5	5-6	5-10	5-14
140000	C	C	O	O	C	6	6-7	6-11	6-15
160000	C	C	O	O	O	7	7-10	7-12	7-16
200000	C	O	C	C	C	10	10-11	10-13	10-17
220000	C	O	C	C	O	11	11-12	11-14	11-20
240000	C	O	C	O	C	12	12-13	12-15	12-21
260000	C	O	C	O	O	13	13-14	13-16	13-22
300000	C	O	O	C	C	14	14-15	14-17	14-23
320000	C	O	O	C	O	15	15-16	15-20	15-24
340000	C	O	O	O	C	16	16-17	16-21	16-25
360000	C	O	O	O	O	17	17-20	17-22	17-26
400000	O	C	C	C	C	20	20-21	20-23	20-27
420000	O	C	C	C	O	21	21-22	21-24	21-30
440000	O	C	C	O	C	22	22-23	22-25	22-31
460000	O	C	C	O	O	23	23-24	23-26	23-32
500000	O	C	O	C	C	24	24-25	24-27	24-33
520000	O	C	O	C	O	25	25-26	25-30	25-34
540000	O	C	O	O	C	26	26-27	26-31	26-35
560000	O	C	O	O	O	27	27-30	27-32	27-36
600000	O	O	C	C	C	30	30-31	30-33	30-37

Table 1 MSV11-D, MSV11-E Addressing Summary (Cont)

Starting	Switch Settings					Memory Bank(s) Selected			
Address	S1-1	S1-2	S1-3	S1-4	S1-5	-DA, -EA	-DB, -EB	-DC, -EC	-DD, -ED
620000	O	O	C	C	O	31	31-32	31-34	X
640000	O	O	C	O	C	32	32-33	32-35	X
660000	O	O	C	O	O	33	33-34	33-36	X
700000	O	O	O	C	C	34	34-35	34-37	X
720000	O	O	O	C	O	35	35-36	X	X
740000	O	O	O	O	C	36	36-37	X	X
760000	O	O	O	O	O	37	X	X	X

NOTES

1. Switch settings

   C = ON
   O = OFF

2. Bank 7 cannot be selected as factory-configured; however, the user can enable the lower 2K portion of bank 7 for use.

3. X = Do not use.

4. Rocker switch positions are defined by pressing the desired side of the rocker, not by the red line on the opposite side of the rocker.

One MSV11-E module draws approximately 7 W when operating in the battery backup mode. A typical backup system that is 30% efficient with a 2.5 ampere-hour battery will support each module for approximately 2 hours. When used in a PDP-11/03 system, or equivalent, no additional cooling of the module is required during the backup period, if the room temperature is maintained to less than 32° C (90° F).

Parity

One jumper is factory-installed for nonparity (MSV11-D) or parity (MSV11-E) operation, depending on model. Do not reconfigure this jumper. Standard jumper configurations are listed below for reference purposes. Refer to the functional description section on parity for recommended implementation.

All MSV11-D models: Jumper installed from pin 7 to pin 5
All MSV11-E models: Jumper installed from pin 6 to pin 5

Memory Size

Two jumpers are factory-installed to configure addressing logic for memory size (number and type of memory integrated circuits). Do not reconfigure these jumpers. Standard jumper configurations are listed below for reference purposes.

	Jumpers (Two Installed)	
Models	**Memory Range Pins**	**Memory Select Pins**
MSV11-DA,-EA	From 17 to 15	From 17 to 14
MSV11-DB,-EB	From 17 to 15	From 12 to 14
MSV11-DC,-EC	From 16 to 15	From 16 to 14
MSV11-DD,-ED	From 16 to 15	From 10 to 14

FUNCTIONAL DESCRIPTION

General

Logic functions and circuits that comprise MSV11-D and MSV11-E memory modules are shown in Figure 2. Both types of memory modules are identical, with the exception that MSV11-E models include parity logic; MSV11-D models do not include parity.

Memory Array

The memory array is the main function contained on the module. Depending on model, the module will contain 16 or 32 dynamic random access memory (RAM) integrated circuits (ICs); MSV11-E models include an additional two or four RAM ICs, depending on model, as part of the parity logic. All RAM ICs will be identical types for a given model.

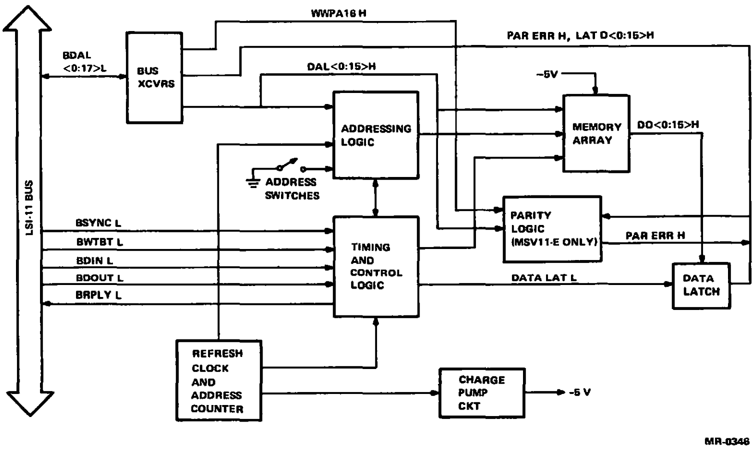


Figure MSV11-D, -E-2
MSV11-D and MSV11-E Logic Functions

Two types are used: 4K by 1 bit and 16K by 1 bit. The number and type of RAMS used are listed for each memory model as follows.

Model	RAM IC Type	Qty RAM ICs in Memory Array	Qty RAM ICs in Parity Logic
MSV11-DA	4K X 1	16	None
MSV11-DB	4K X 1	32	None
MSV11-DC	16K X 1	16	None
MSV11-DD	16K X 1	32	None
MSV11-EA	4K X 1	16	2
MSV11-EB	4K X 1	32	4
MSV11-EC	16K X 1	16	2
MSV11-ED	16K X 1	32	4

The memory array is organized as shown in Figure 3. As previously listed, either 16 or 32 memory ICs comprise the memory array. Models that include only 16 ICs are organized with 8 ICs in the high byte and 8 ICs in the low byte of the low range array shown on the figure. Models that include 32 ICs include the 16 ICs described for the 16-IC models, plus an additional 16 ICs for the high and low bytes of the high range array. Row address strobe (RAS0 L and RAS1 L) control signals, produced by the addressing logic, select the appropriate low or high range array. Write byte (WTBT1 L and WTBTO L) control signals are produced by the addressing logic during a memory write operation to select the addressed byte (DATOB bus cycle); during a word write operation (DATO bus cycle), WTBT L is high and selects both bytes.

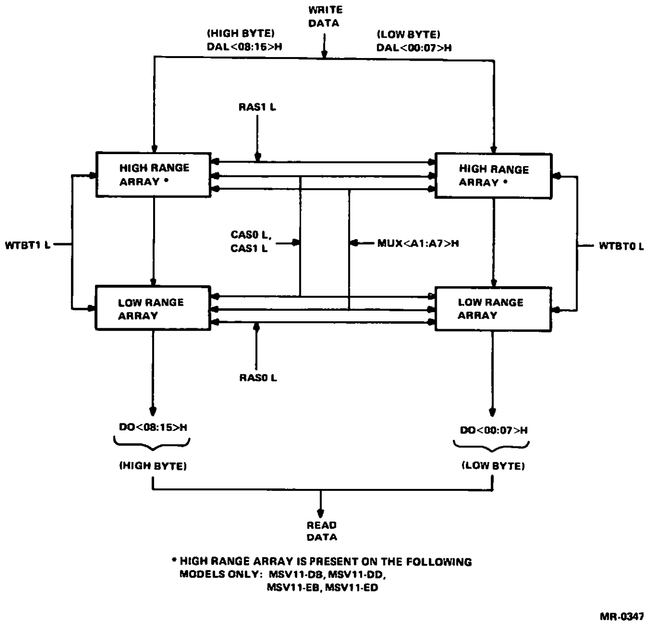


Figure MSV11-D, -E-3
Memory Array

Fourteen address bits are required for 16K by 1 bit RAM ICs, and 12 address bits are required for 4K by 1 bit RAMs. The required 12- or 14-bit address is multiplexed over 6 or 7 address lines [MUX (A1:A7)H] to all RAM ICs that comprise the memory array. Addressing is controlled by column address strobe (CASO L and CAS1 L) and row address strobe (RASO L and RAS1 L) signals.

Addressing Logic

Addressing logic (Figure 4) receives addresses from the LSI-11 bus and produces address bits and control signals when the received address is for a memory location on the module. The user configures a starting

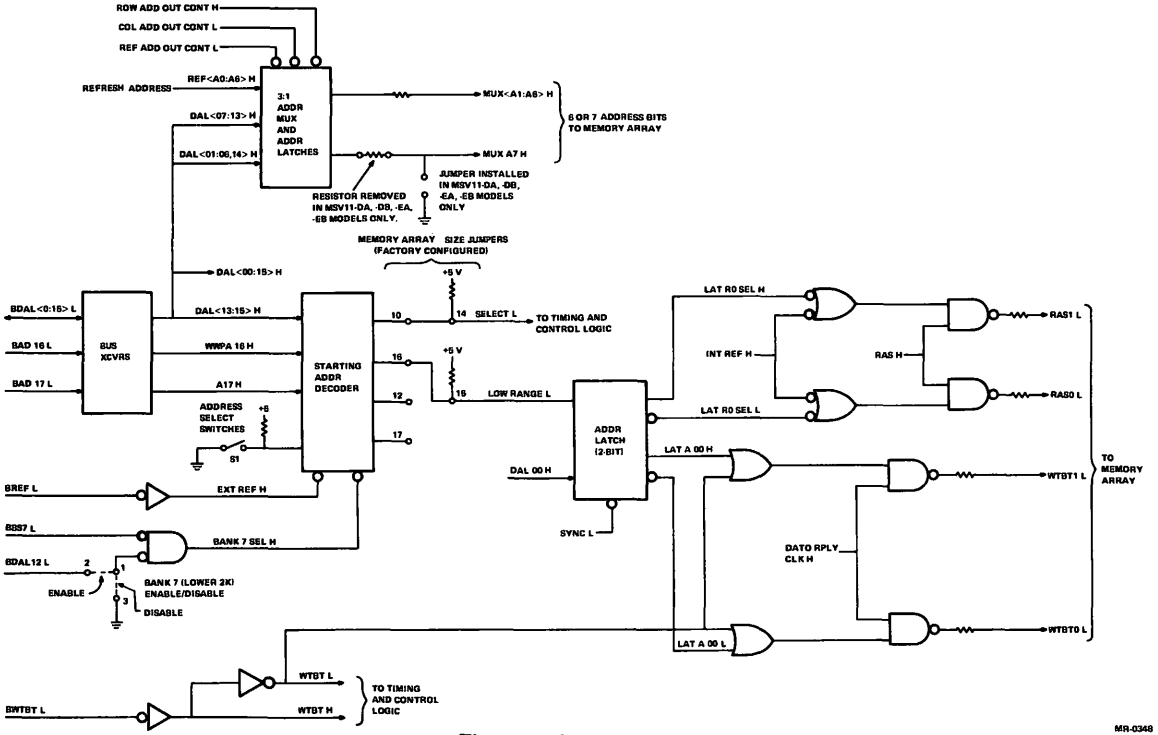


Figure MSV11-D, -E-4
Addressing Logic

address (described in Table 1) that defines the lowest address in the module's range. When an address is received that resides in the module's user-configured range, SELECT L goes active. SELECT L is produced by decoding address bits DAL (13:15) H and the starting address configured by S1–S5. Memory array size jumpers select the appropriate SELECT L and LOW RANGE L decoder ROM outputs, depending on MSV11-D or MSV11-E model; these jumpers are factory-configured and normally should not be changed. SELECT L initiates the memory cycle in the timing and control logic. LOW RANGE L controls selection of RAS0 L or RAS1 L signals that select the low range array for all models, or the high range array when addressed on MSV11-DB, -DD, -EB, and -ED models.

The starting address decoder is always inhibited during external refresh operations; EXT REF H goes low during normal memory access operations.

The upper 4K address space in all systems is normally reserved for peripheral devices. However, the user can enable the use of the lower 2K portion of bank 7 by installing a jumper. When bank 7 is not enabled, BBS7 L is inverted by a bus receiver producing BANK 7 SEL H; the high signal inhibits the starting address decoder ROM and no active outputs are produced. When the lower 2K portion of bank 7 is enabled, a jumper on the input of the BBS7 L bus receiver is reconfigured to inhibit the bus receiver when BDAL 12 L is high (the lower 2K portion). Thus, BANK 7 SEL H will go high only when an address resides in the upper 2K portion of bank 7 and inhibits memory address decoding.

MSV11-D and MSV11-E models will not respond to external refresh signals, only to "on-board" refresh described in the paragraph entitled "Memory Refresh." When the externally generated BREF L signal is asserted, EXT REF H goes high and inhibits the starting address decoder ROM.

Timing and Control Logic
Circuits comprising timing and control logic are shown in Figure 5. Signal sequences for DATI, DATO(B), DATIO(B), and refresh cycles are shown in Figures 6 through 9, respectively.

The major portions of timing and control functions are contained in the timing and refresh arbitration logic (Figure 5). Basic timing for any memory cycle is produced by a tapped delay line and appropriate gating logic. Additional logic functions arbitrate refresh cycles, produce control signals for the memory array, addressing logic, and parity logic functions, and generate appropriate BRPLY L signals during any memory access operation.

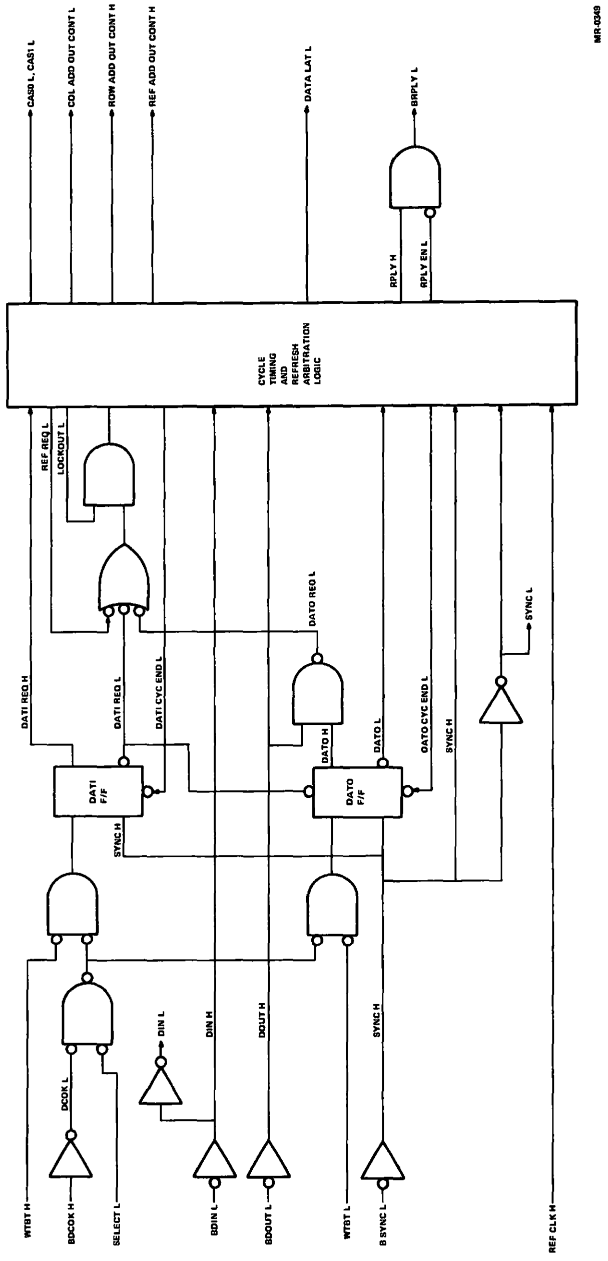


Figure MSV11-D, -E-5
Timing and Control Logic

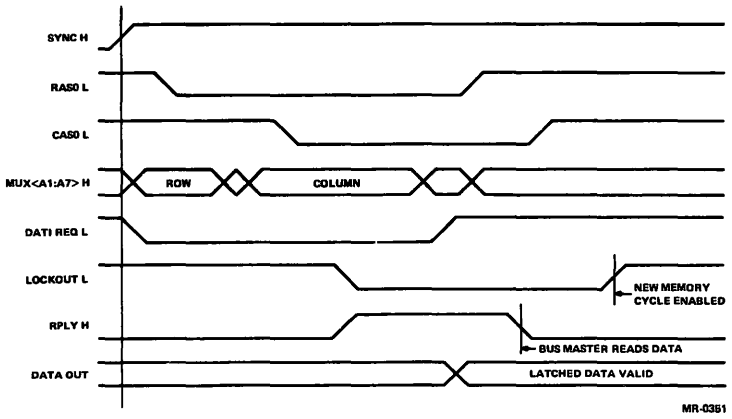


Figure MSV11-D, -E-6
DATI Signal Sequence

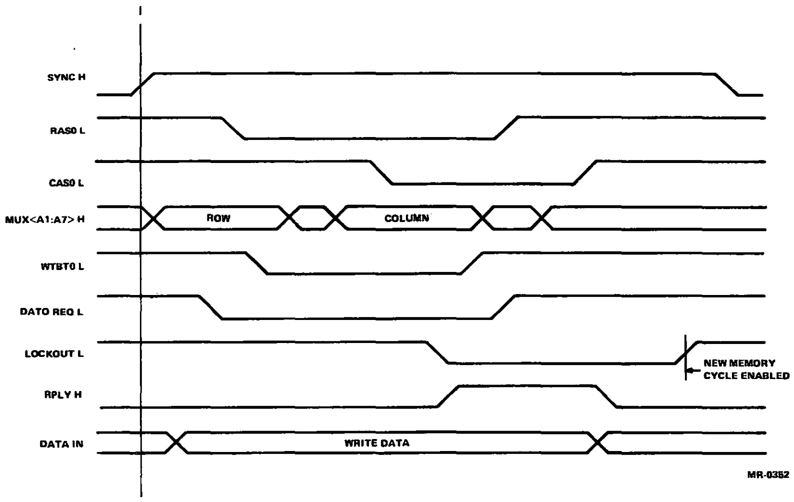


Figure MSV11-D, -E-7
Figure 7 DATO(B) Signal Sequence

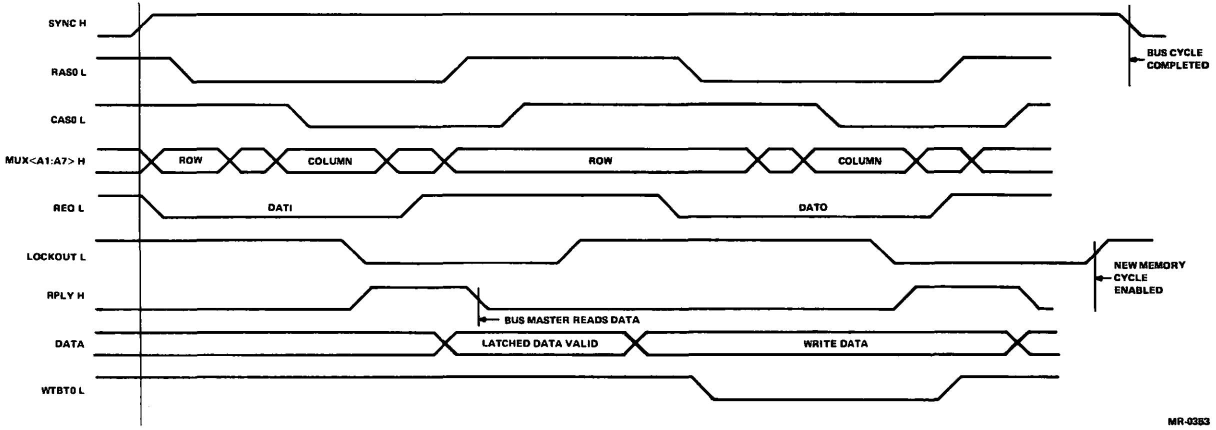


Figure MSV11-D, -E-8
DATIO(B) Signal Sequence

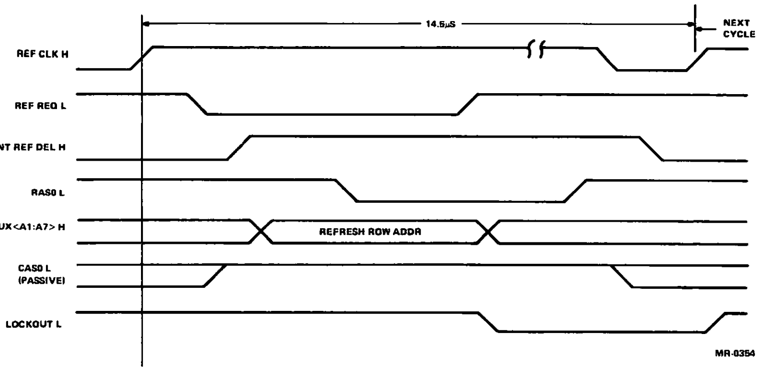


Figure MSV11-D, -E-9
Memory Refresh Signal Sequence

A memory cycle (other than refresh) is initiated by the active SELECT L signal produced by addressing logic. The leading edge of SYNC H clocks either the DATI or DATO flip-flop to the set state, depending on the state of WTBT H and WTBT L; these two signals, produced by the LSI-11 bus BWTBT L signal, go active during the addressing portion of the cycle only when a DATO(B) cycle is in progress; DATO H enables DATA REQ L when DOUT H goes active (high). The appropriate DATI REQ L or DATO REQ L signal is ORed with REF REQ L, producing a high signal that initiates the timing sequence.

If a refresh cycle is in progress when the DATI, DATO(B), or DATIO(B) cycle is initiated, the refresh operation is first completed before continuing the memory access operation; LOCKOUT L goes active during any cycle timing sequence and inhibits the new request from starting another cycle until the present cycle has been completed. However, if a memory access cycle is initiated (addressing portion completed) and a refresh cycle request occurs, refresh arbitration logic delays the start of the memory access cycle approximately 100 ns. If a refresh conflict occurs (refresh wins), the refresh cycle will be first completed and add approximately 575 ns to the DATI or DATO(B) cycle time. If memory has been accessed (LOCKOUT L goes passive), a refresh cycle can be initiated although the bus cycle "handshaking" may not have been completed.

A DATIO(B) bus cycle is similar to a DATI cycle followed by a DATO(B) cycle; however, only the addressing portion of the cycle prior to the DATI portion occurs, according to LSI-11 bus protocol.

Write-byte operations (DATOB or the DATOB portion of DATIOB cycles) are controlled by the bus BWTBT L signal and the addressing logic. The timing and control functions are exactly the same for write byte or write word operations.

Memory Refresh

Memory refresh request logic is shown in Figure 10. A 14.5 μs refresh clock operates the refresh request time to allow completion of 128 refresh cycles during any 2 ms period. A refresh address counter increments once on each refresh cycle, producing the current refresh row address. Sequential row addresses are thus refreshed, completing all 128 rows within 2 ms for 16K by 1 bit memory integrated circuits, or 64 rows within 1 ms for 4K by 1 bit memory integrated circuits.

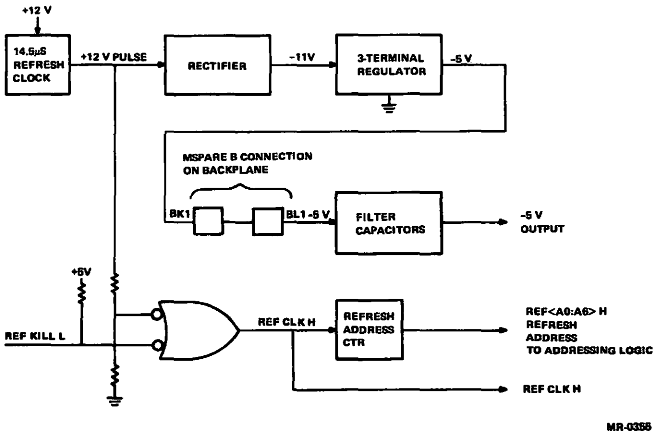


Figure MSV11-D, -E-10
Refresh Logic and Charge Pump Circuit

Charge Pump Circuit

The charge pump circuit (Figure 10) produces −5 Vdc for the memory array. Input power is obtained from the +12 V system or battery backup power applied via the refresh clock. The resulting 12 V 14.5 μs refresh clock pulse is applied to a rectifier circuit which produces a −11 Vdc (approximately) output. A 3-terminal regulator then produces the required regulated −5 Vdc.

Note that the −5 V is applied to the filter capacitors via MSPAREB backplane pins. This is done for manufacturing test purposes. These pins are connected on all LSI-11 backplanes as shown on the figure. If nonstandard backplanes (user-supplied) are used, be certain that these pins are connected.

MSV11-E Parity Logic

MSV11-E parity logic functions are shown in Figure 11. The basic functions include a parity generator for memory write data, a 2-bit memory array, a parity detector, and a latch circuit. The parity generator produces two parity bits, one for each main memory byte. Address and control signal (not shown) lines are identical to those applied to the main memory array. When any main memory location is read, the corresponding

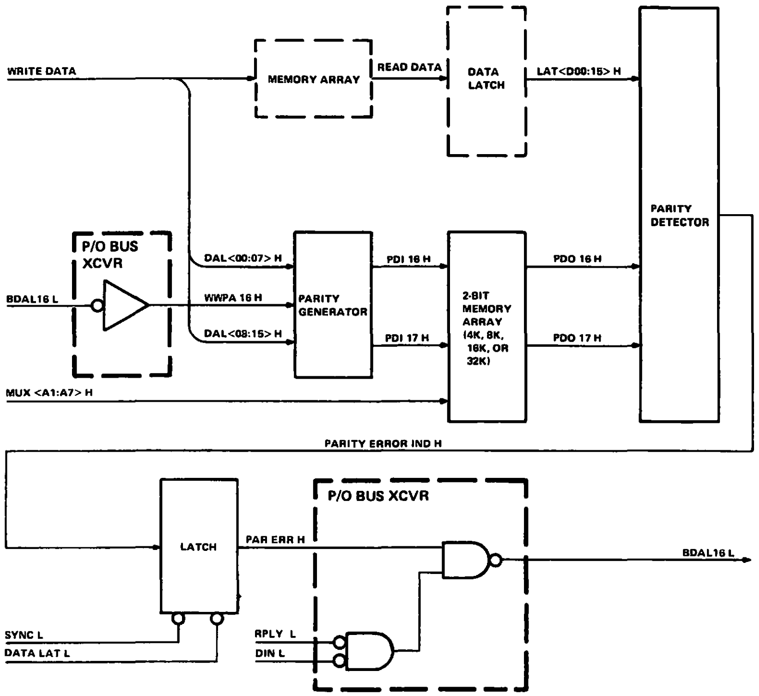


Figure MSV11-D, -E-10
Parity Logic

parity memory location is read. Parity bits are checked by the parity detector and the result is stored in the output latch. If a parity error is detected (PAR ERROR IND H is active), PAR ERR H is gated onto the BDAL 16 L bus line. The processor reads the error during the memory read (DATI) cycle and responds accordingly.

MSV11-E memory modules respond to bus master devices by delaying BRPLY L assertion to accommodate the parity generator and parity detector logic. This timing is jumper-selected (factory-configured) and should not be changed.

Parity logic functions are tested when running memory diagnostics by writing incorrect parity bits. The processor forces this condition during a DATO(B) cycle by asserting BDAL 16 L during the output data transfer portion of the bus cycle. BDAL 16 L is received, inverted to produce "write wrong parity" (WWPA 16 H), and applied to the parity generator, forcing it to store incorrect parity bits. The error is then detected by subsequent memory read cycles.

LSI-11 processors do not support the parity implementation used on the MSV11-E. However, the user may implement logic to use the PAR ERR signal that is gated onto the BDAL 16, and to assert it for testing the parity function. A recommended method could be a module that clocked BDAL 16 on the trailing edge of DIN. Any time BDAL 16 is asserted, a parity error has occurred. The user can then cause an interrupt, increment a counter, etc. Note that the instruction execution cannot be aborted under this scheme, as is the normal procedure when detecting a parity error.

MEMORY AND ASYNCHRONOUS SERIAL LINE INTERFACE

GENERAL

The MXV11-A is a dual height multifunction option module for the LSI-11, LSI-11/2 or LSI-11/23. It contains a read/write memory, provisions for read-only memory, two asynchronous serial line interfaces and a 60 Hz clock signal derived from a crystal oscillator. Read/write memory is supplied with either 8K or 32K bytes (4K or 16K words). Two 24-pin sockets are provided for +5 V read-only memories. 1K × 8, 2K × 8, or 4K × 8 ROMS may be used. The sockets may also be used for 256 words of bootstrap code.

The two asynchronous serial lines transmit and receive EIA-423 signal levels from 150 baud to 38.4K baud. 20 mA active or passive current loop operation at 110 baud may be obtained with the DLV11-KA EIA to 20 mA converter option. The serial lines will not support the reader run function of the DLV11-KA option. The serial lines provide error indicator bits for overrun error, frame error, and parity error, but do not have modem controls. Serial line 1 may be configured to respond to a break signal. The serial lines have single level interrupt logic and should be placed after multi-level devices on an LSI-11/23 system. Serial line 1 may be used as a console port, or, along with serial line 0, may be used with any of several standard types of serial communication devices.

The 60 Hz clock signal can be selected by a wirewrap jumper to provide line time clock interrupts on the bus.

FEATURES

- 8K or 32K bytes of RAM with on-board refresh
- User-configured with choice of PROM or system device bootstrap ROM option (MXV11-A2)
- Accepts two 5V, 24-pin UVPROM or fusible-link PROM chips
- Two serial lines meeting RS-423 standard (backward compatible with RS-232C), baud rates up to 38.4K
- Jumper selected crystal controlled baud rates of 150, 300, 1.2K, 2.4K, 4.8K, 9.6K, 19.1K, 38.4K baud, and external.
- Crystal clock—60Hz

Model Designations

MXV11-AA	LSI-11 multifunction module, 8K-byte Random Access Memory.

MXV11-A

MXV11-AC	LSI-11 multifunction module, 32K-byte Random Access Memory

SPECIFICATIONS

Identification	M8047-AA	MXV11-AA	8K bytes
	M8047-BA	MXV11-AC	32K bytes

Size	Double
Power	+5V, 1.2 A +12V, 0.1 A
Bus Loads	
AC	2
DC	2

RAM Performance

Bus Cycle Type	T acc (ns) Typical	Max
DATI	280	300
DATO(B)	395	410

NOTES:

1. Access time (T acc) is from SYNC to RPLY
2. Assumes memory is not busy and there is no arbitration
3. Refresh arbitration adds 100 ns typical and 120 ns maximum to access time.
4. Refresh conflict adds 575 ns typical and 600 ns maximum to access time.
5. Assumes that SYNC to DOUT time = 285 ns.

ROMs

Power:	+5 V ±5%
Pins:	24 Pin Spacing
Access Time:	Up to 450 nanoseconds
Array Size:	1K × 8, 2K × 8, or 4K × 8 bits
Type:	See accompanying tables

Typical PROM Types

UV PROMS	Chip Array Size	Memory Size
Intel 2758	1K × 8 bits	1K words
Intel 2716	2K × 8 bits	2K words
Intel 2732	4K × 8 bits	4K words
Mostek MK2716	2K × 8 bits	2K words
T.I. TMS 2516	2K × 8 bits	2K words
T.I. TMS 2532	4K × 8 bits	4K words
Bipolar PROMS		
Intel 3628	1K × 8 bits	1K words
Signetics 82S 2708	1K × 8 bits	1K words
Signetics 82S 181	1K × 8 bits	1K words
Signetics 82S 191	2K × 8 bits	2K words

CONFIGURATION

General

The following paragraphs describe how the user can configure the module so that it will function in the system. The jumpers used on this module are of two types. Two jumpers consist of insulated wires soldered to plated-through holes, and the remaining jumpers are wirewrap pins to which connections are made. The soldered jumpers are factory configured and should not be changed. When installing jumpers, the wire runs must be arranged so no more than two wires are on each post and there is no level jumping between posts.

The ROM and RAM memories should not be configured to cover the same area of memory. There is no overlay protection logic to prevent conflicts in this case. The RAM memory will not respond to addresses in the I/O page area (bank 7 in 16-bit address systems). This prevents conflicts when peripherals (including the on-board SLUs) are addressed.

Factory Configuration Wiring

Function	Wire Wrap Pins From	To
RAM Bank 0	J30	J31
	J32	J33
	J31	J32
SLU Channel 0 Address 176500	J23	J18
	J24	J19

			MXV11-A
SLU Channel 1 Address 177560		J28	J19
		J26	J15
		J25	J14
		J27	J13
ROM Bootstrap (TU58)		J37	J38
		J21	J22
		J34	J37
		J33	J39
		J29	J15
SLU Vectors	CH0 (300)	J53	J57
	CH1 (60)	J54	J52
		J55	J54
		J56	J51
SLU Parameters (8 Data bits, no parity, one stop bit)		J59	J61
		J62	J64
		J60	J63
		J61	J62
		J59	J66
		J63	J65
Baud Rates	CH0 (38.4K)	J45	J50
	CH1 (9600)	J46	J48
Break Generation (Halt option)		J6	J7
Crystal Clock		J68	J67

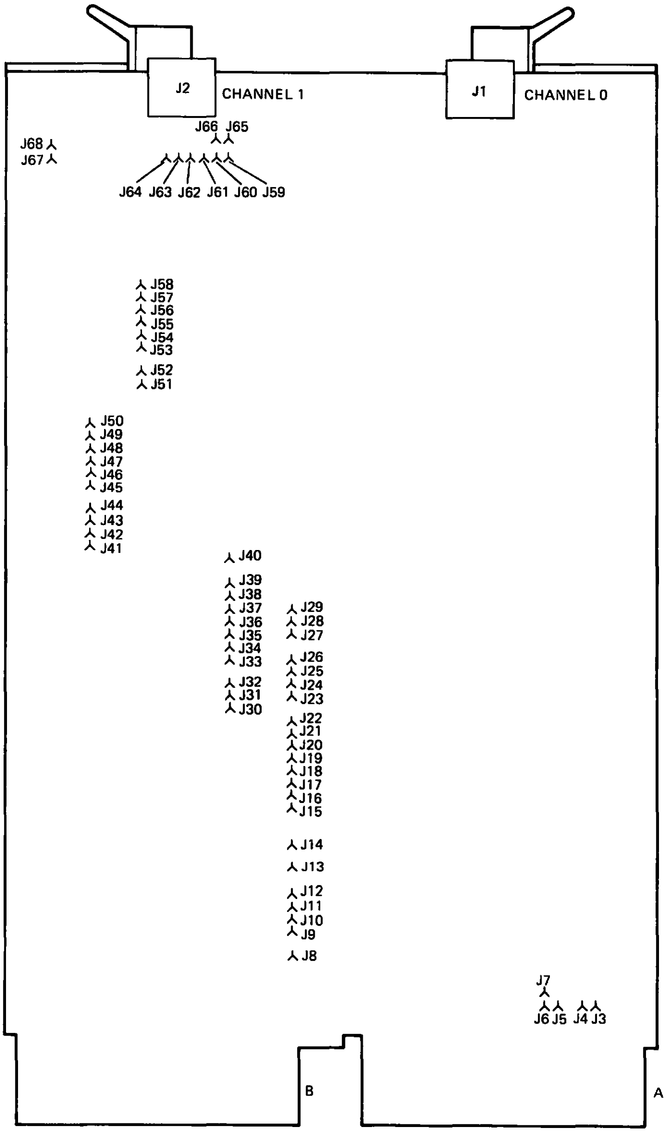


Figure MXV11-A-1
MXV11-A Jumper Locations

MXV11-A Jumper Functions

Pin	Function	Option
J3	Clock L. Open collector output of the clock. Connected to Pin AF1 (SSpare 2). Wirewrap to J4 to implement the clock option.	60 Hz
J4	BEVNT L. Event interrupt (Pin BR1) used for the clock option.	60 Hz
J5	BDCOK H. DCOK (Pin BA1) when HIGH allows the processor to operate, when LOW initializes the system. Connected to J6 to implement the boot option.	BOOT
J6	Framing error. Open collector output of framing error from serial line one. Connected to Pin AE1 (SSpare 1). Wirewrap to J5 to implement the boot option or to J7 for the HALT option. Reset by bus initialize or reception of a valid character.	BREAK
J7	BHALT L. Halt (Pin AP1) when LOW will stop program execution and cause the processor to enter ODT microcode. Connected to J6 to implement the Halt option.	HALT
J8	GND. A ground signal that can be used to disable ROM by wirewrapping to J21 or to disable a serial line by wirewrapping to an address input pin (J23 or J24 for serial line 0; or J25, J26, J27, or J28 for serial line 1).	ROM
J9	A13 L. Address bit 13 asserted LOW. Wirewrap to J11 to select Bank 1 with the ROM address decoder.	ROM

Pin	Function	Option
J10	A13 H. Address bit 13 asserted HIGH. Wirewrap to J11 to select Bank 0 with the ROM address decoder.	ROM
J11	A13 M. Address bit 13 input to the ROM address decoder. See J9 and J10. Used only if J20 is wirewrapped to J21.	ROM
J12	A03 H. Address bit 03 asserted HIGH. Wirewrapped to the serial line address decoders (J23 or J24 for serial line 0; J25, J26, J27 or J28 for serial line 1) when address bit 03 is to be decoded as a 1.	SLU
J13	A04 H. Address bit 04 asserted HIGH. Wirewrapped to the serial line address decoders when address bit 04 is to be decoded as a 1.	SLU
J14	A05 H. Address bit 05 asserted HIGH. Wirewrapped to the serial line one address decoder when address bit 05 is to be decoded as a 1.	SLU
J15	A09 H. Address bit 09 asserted HIGH. Wirewrapped to the serial line one address decoder when address bit 09 is to be decoded as a 1.	SLU
J16	A09 L. Address bit 09 asserted LOW. Wirewrapped to the serial line one address decoder when address bit 09 is to be decoded as a 0.	SLU
J17	A05 L. Address bit 05 asserted LOW. Wirewrapped to the serial line one address decoder when address bit 05 is to be decoded as a 0.	SLU

Pin	Function	Option
J18	A04 L. Address bit 04 asserted LOW. Wirewrapped to the serial line address decoders when address bit 04 is to be decoded as a 0.	SLU
J19	A03 L. Address bit 03 asserted LOW. Wirewrapped to the serial line address decoders when address bit 03 is to be decoded as a 0.	SLU
J20	ROM address. Output of the ROM address decoder. Connected to J21 when ROM is to be used in Bank 0 or Bank 1.	ROM
J21	ROM select. ROM address selection enable asserted HIGH. Wirewrapped to J8 (GND) to disable ROM, to J20 for Bank 0 or Bank 1, or to J22 for bootstrap.	ROM
J22	Boot address. Output of the bootstrap address decoder. Connected to J21 when ROM is to be used in the bootstrap range from 173000—173776 (773000—773776 for LSI-11/23).	BOOT
J23	Serial line zero address decoder input asserted HIGH. May be wirewrapped to A03 H (J12), A03 L (J19), A04 H (J13) or A04 L (J18).	SLU
J24	Serial line zero address decoder input asserted HIGH. May be wirewrapped to A03 or A04, whichever bit is not wired to J23. May be wirewrapped to GND (J8) to disable serial line zero.	SLU

Pin	Function	Option
J25 through J28	Serial line one address decoder input asserted HIGH. Four address decoder inputs to be connected to address bits A03, A04, A05, and A09. Whether the HIGH or LOW assertion state of a bit is wirewrapped to an input determines if that bit is decoded as a 1 or a 0. See J12 through J19. May be wirewrapped to GND (J8) to disable serial line one.	SLU
J29	ROM address bit 09 input. Wirewrapped to A09 H (J15) for normal ROM addressing and also for the MXV11-A2 option when the TU58 bootstrap is desired. Wirewrapped to A09 L (J16) for the MXV11-A2 option when the disk bootstrap is desired.	ROM
J30 through J32	RAM starting address selection. These pins are wirewrapped to J33 (logic 0) or J34 (logic 1) to select the RAM starting address. (See below)	RAM

J32	J31	J30	Bank	Starting Address
0	0	0	0	000000
0	0	1	1	020000
0	1	0	2	040000
0	1	1	3	060000
1	0	0	4	100000
1	0	1	5	120000
1	1	0	6	140000
1	1	1	7	160000

Pin	Function	Option
J33	GND. Logic 0 level signal used for selecting the RAM starting address and for enabling some ROM ICs in the ROM sockets.	RAM, ROM

Pin	Function	Option
J34	+3 V. Logic 1 level signal used for selecting the RAM starting address and for enabling some ROM ICs in the ROM sockets.	RAM, ROM
J35	A12 H. Address bit 12 asserted HIGH. Used for addressing 4K × 8 bit ROMs. Wirewrapped to J37, J38 or J39, depending on the ROM used.	ROM
J36	A11 H. Address bit 11 asserted HIGH. Used for addressing 2K × 8 and 4K × bit ROMs. Wirewrapped to J37, J38 or J39, depending on the ROM.	ROM
J37	Pin 18 on both ROM sockets. Used for addressing or enabling ROM. Wirewrapped to J33 for ground, to J34 for +3 V, to J35 for A12 or to J36 for A11.	ROM
J38	Pin 19 on both ROM sockets. Used for addressing or enabling ROM. Wirewrapped to J33 for ground, to J34 for +3 V, to J35 for A12 or to J36 for A11.	ROM
J39	Pin 21 on both ROM sockets. Used for addressing or enabling ROM. Wirewrapped to J33 for ground, to J34 for +3 V, to J35 for A12, to J36 for A11 or to J40 for +5 V.	ROM
J40	+5V. Used to power some ROMs on pin 21.	ROM
J41	Used for 150 baud. Wirewrapped to J45 for serial line 0, to J46 for serial line 1. (See accompanying table)	SLU
J42	Used for 1200 baud.	SLU
J43	Used for 300 baud.	SLU

Pin	Function	Option
J44	Used for 2400 baud.	SLU
J45	Clock 0. The clock input for serial line 0 transmit and receive, 16 times the baud rate. Wirewrapped to either J41, J42, J43, J44, J47, J48, J49, or J50.	SLU
J46	Clock 1. The clock input for serial line 1 transmit and receive, 16 times the baud rate. Wirewrapped to either J41, J42, J43, J44, J47, J48, J49 or J50.	SLU
J47	Used for 4800 baud.	SLU
J48	Used for 9600 baud.	SLU
J49	Used for 19.2K baud.	SLU
J50	Used for 38.4K baud.	SLU
J51	Vec 0. Vector enable for Channel 0. Used to drive vector bits that pass the test: logic 1 for channel 0 and logic 0 for channel 1. Wirewrapped to J53 for bit 03, to J54 for bit 04, to J55 for bit 05, to J56 for bits 06 and 07.	SLU
J52	Vec 1. Vector enable for Channel 1. Used to drive vector bits that pass the test: logic 0 for Channel 0 and logic 1 for Channel 1. Wirewrapped to J53 for bit 03, to J54 for bit 04, to J55 for bit 05, to J56 for bits 06 and 07	SLU
J53	Vector bit 03. Selects how bit 03 is to be driven for interrupt vectors. Wirewrapped to J51 if a logic 1 for Channel 0 and a logic 0 for Channel 1, to J52 if a logic 0 for Channel 0 and a logic 1 for Channel 1, to J57 if a logic 0 for both Channel 0 and Channel 1, or to J58 if a logic 1 for both Channel 0 and Channel 1.	SLU

Pin	Function	Option
J54	Vector bit 04. Selects how bit 04 is to be driven for interrupt vectors. Wirewrapped the same as J53.	SLU
J55	Vector bit 05. Selects how bit 05 is to be driven for interrupt vectors. Wirewrapped the same as J53.	SLU
J56	Vector bits 06 and 07. Selects how bits 06 and 07 are to be driven for interrupt vectors. Wirewrapped the same as J53.	SLU
J57	GND. Logic 0 signal for configuring vector bits. Wirewrapped to J53, J54, J55 and/or J56 when the corresponding vector bit(s) will be logical 0 for both serial line channels.	SLU
J58	+3 V. Logic 1 signal for configuring vector bits. Wirewrapped to J53, J54, J55 and/or J56 when the corresponding vector bit(s) will be a logical 1 for both serial line channels.	SLU
J59	7 bits parity/8 bits no parity, Channel 1. Wirewrapped to ground (J65) for seven bits with parity or to +3 V (J66) for eight bits with no parity.	SLU
J60	2 stop bits. Selects one or two stop bits for Channel 1. Wirewrapped to ground (J65) for one stop bit or to +3 V (J66) for two stop bits.	SLU
J61	Even parity. Selects odd or even parity for Channel 1 when seven bits with parity (J59 wirewrapped to ground) is selected. Wirewrapped to ground	SLU

Pin	Function	Option
	(J65) for odd parity or to +3 V (J66) for even parity.	
J62	7 bits parity/8 bits no parity, Channel 0. Wirewrapped to ground (J65) for seven bits with parity or to +3 V (J66) for eight bits with no parity.	SLU
J63	2 stop bits. Selects one or two stop bits for Channel 0. Wirewrapped to ground (J65) for one stop bit or to +3 V (J66) for two stop bits.	SLU
J64	Even parity. Selects odd or even parity for Channel 0 when seven bits with parity (J59 wirewrapped to ground) is selected. Wirewrapped to Logic 0 (J65) for odd parity or to Logic 1 (J66) for even parity.	SLU
J65	Logic zero. Ground signal used for configuring serial line interfaces.	SLU
J66	Logic one. +3 V signal used for configuring serial line interfaces.	SLU
J67	Clock in. Clock input for baud rates, memory refresh and negative voltage generator. Wirewrapped to J68. Not a user option.	SLU
J68	Clock out. Crystal oscillator output at 19.6608M Hz. Wirewrapped to J67. Not a user option.	SLU

Configuring the RAM

The RAM can be configured to start on any 8KB boundary below 64KB. Because of this restriction, the MXV11 (8KB version) is not usable for memory above 56KB. The MXV11 can be used in 18-bit memory address systems, but it is restricted to being assigned to the memory area below 56KB.

Five wirewrap terminals, J30 through J34, select the starting address. The figure shows the jumper configurations required to obtain the desired starting addresses.

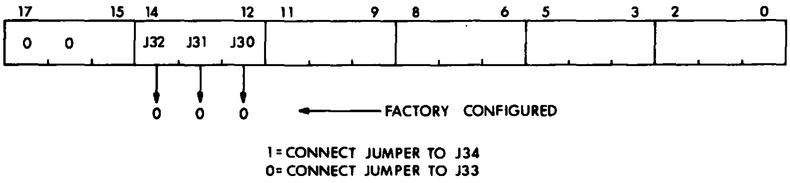


Figure MXV11-A-2
RAM Starting Address Selection

Configuring the ROM
Depending on the ROM type, the module's capacity is 1K, 2K, or 4K words using a pair of 1024 × 8, 2048 × 8 or 4096 × 8-bit ROMs respectively. The user configures jumpers on the module for the ROM type being used. The actual procedure for loading data into EPROMS, PROMS (or writing specifications for masked ROMs) will vary depending on the manufacturer, and are beyond the scope of this section. The user must refer to the manufacturer's data sheets and to the Chapter *Using PROMs*. The user must be aware of the relationship of the EPROM, PROM or ROM pins to the LSI-11 data bits, and the relationship of the pins to the memory address bits. Refer to the accompanying figure for ROM socket pin assignments. All ROMs used on the MXV11-A must conform to these pin assignments.

The factory configuration allows for using the MXV11-A2 bootstrap ROMS.

Configuring the Bootstrap ROM — The ROM can be configured to operate in the I/O page to support bootstrap programs. The address area contains 256 words from 173000 to 173776 (773000 to 773776 for the LSI-11/23.

The MXV11-A is configured at the factory to allow for using the MXV11-A2 TU58 bootstrap. To reconfigure to use the disk bootstrap of the MXV11-A2, remove jumper J29 to J15 and install jumper J29 to J16.

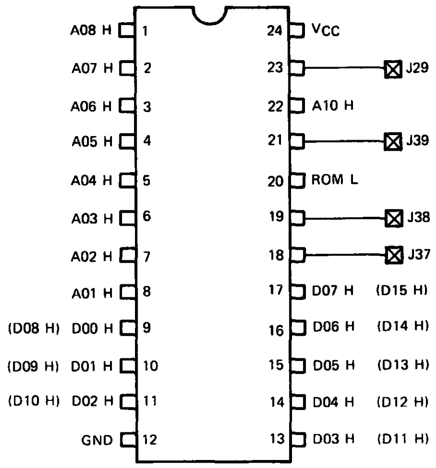
ROM Bank Selection — If the MXV11-A sockets are used for program ROM instead of a bootstrap ROM, the memory must be selected by a jumper connecting J20 to J21. When main ROM memory is selected, the entire 4K word bank in enabled. If a 1K or 2K ROM is used, it will "wrap-around" and give invalid data depending on how the address lines are configured when the nonexisting ROM area is addressed. Main memory may be positioned in bank 0 or bank 1. To position the ROM in bank 0, jumper J10 to J11. To position the ROM in bank 1, jumper J9 to J11. These jumper functions are described in the accompanying table.

Configuring for Specific ROM Types — Additional jumpers must be connected depending on the type of ROM used. The accompanying table describes the jumper configuration when using typical ROMs such as the Intel 2716 (2K × 8) or 2732 (4K × 8) EPROMs. The user must refer to the manufacturer's data sheets when configuring jumpers for other ROM types.

The function of wirewrap pins J29, J38, J37 and J39 are shown in the accompanying figure. These pins are to be connected as required to pins J33 through J40.

EPROM Address Jumpers

		2716 ROM		**2732 ROM**	
Function		**Bank 0**	**Bank 1**	**Bank 0**	**Bank 1**
	From	**To**	**To**	**To**	**To**
Bank Enable	J20	J21	J21	J21	J21
Bit 09 Input	J29	J15	J15	J15	J15
Address or Enable	J38	J36	J36	J36	J36
Address or Enable	J37	J33	J33	J35	J35
Address or Enable	J39	J40	J40	J33	J34



NOTE:
DATA OUT PINS SHOWN IN PARENTHESES REFER TO THE HIGH BYTE SOCKET XE67.
DATA OUT PINS D00 H THROUGH D07 H REFER TO THE LOW BYTE SOCKET XE57.

Figure MXV11-A-3
MXV11-A ROM Socket Pin Assignment

CONFIGURING THE SERIAL LINE UNITS.

Serial Line Register Address Selection

Four device registers (RCSR, RBUF, XCSR and XBUF) are provided for each of the two serial lines. Jumpers are configured to establish separate base addresses for each serial line as shown.

- Serial port 0 may be assigned to one of four starting addresses: 176500, 176510, 176520, 176530.
- Serial port 1 may be assigned addresses in two ranges. The first range starts at 176500 and covers the eight starting addresses from 176500 to 176570. The second range starts at 177500 and also contains eight possible starting addresses, including the standard console address, 177560. Since several other standard DIGITAL devices use addresses in this second range, it is recommended that only the console address be used.

The format of an SLU address is shown in the accompanying figure. Note that bits 13-17 are neither configured on nor decoded by the MXV11-A module. These bits are decoded by the bus master module as the bank 7 select (BBS7 L) bus signal. This signal becomes active only when the I/O page is accessed. Bit 0 is used as the byte pointer.

Bits 1 and 2 select one of the four device registers within the addressed serial line. Bits 3 and 4 are used to select one of four possible device addresses for serial line 0. Bits 3, 4, 5 and 9 are used to select the device addresses in two ranges for serial line 1 (console). The table describes the jumper combinations to select one of four device addresses for serial line 0 (I/O).

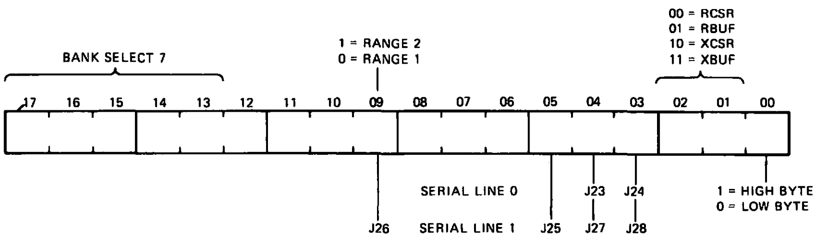
Serial Line 0 Address Jumpers

Address	**Jumper Posts**	
(Octal)	**J23 to**	**J24 to**
176500	J18 (Logic 0)	J19 (Logic 0) Factory Configuration
176510	J18 (Logic 0)	J12 (Logic 1)
176520	J13 (Logic 1)	J19 (Logic 0)
176530	J13 (Logic 1)	J12 (Logic 1)

NOTE

Logic 1	Logic 0
J13 (A04 H)	J18 (A04 L)
J12 (A03 H)	J19 (A03 L)

Serial line 1 may have 16 possible device addresses in two ranges. The table describes the jumper combinations to select the eight device registers available in range 1. Only one device address is used in range 2.



NOTE:
JUMPER POSTS ARE WIRED TO A HIGH ADDRESS LINE FOR A 1 AND TO A LOW ADDRESS LINE FOR A 0. REFER TO TABLES 7 AND 8 FOR JUMPER CONFIGURATIONS.

Figure MXV11-A-4
MXV11-A SLU Address Format

Serial Line 1 Address Jumpers

Address (Octal) Range 1	**Jumper Posts** J26 to	J25 to	J27 to	J28 to
176500	J16	J17	J18	J19
176510	J16	J17	J18	J12
176520	J16	J17	J13	J19
176530	J16	J17	J13	J12
176540	J16	J14	J18	J19
176550	J16	J14	J18	J12
176560	J16	J14	J13	J19
176570	J16	J14	J13	J12
Range 2				
177560 (See note)	J15	J14	J13	J19

NOTE

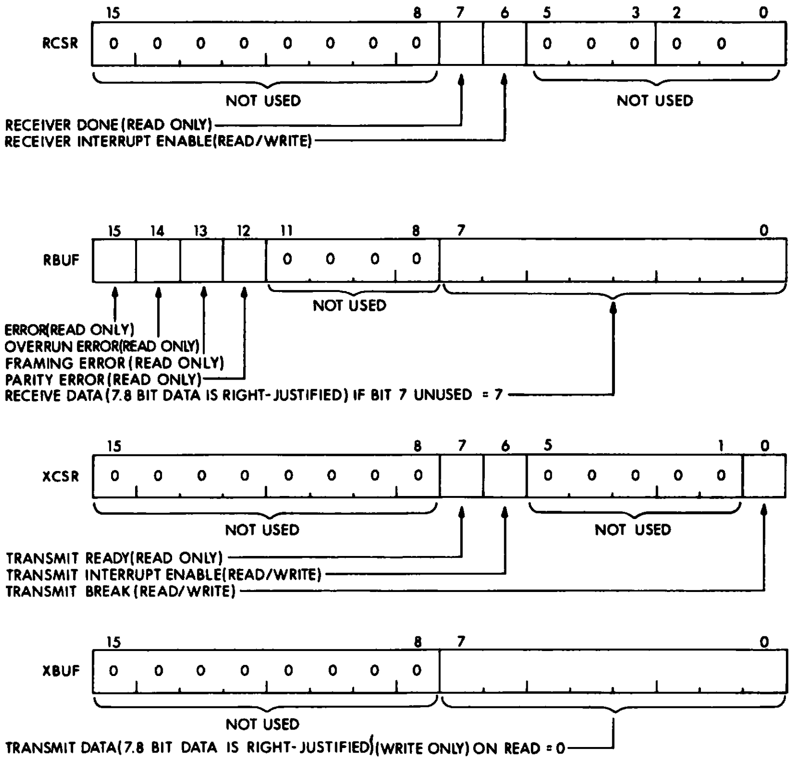
Factory configurations use only one address in range 2 to avoid possible device conflicts. The remaining addresses are pre-assigned to other devices.

Logic 1	Logic 0
J15 (A09 H)	J16 (A09 L)
J14 (A05 H)	J17 (A05 L)
J13 (A04 H)	J18 (A04 L)
J12 (A03 H)	J19 (A03 L)

Control/Status Register

The MXV11-A has two control/status registers (CSRs) for each of its two serial line units. The figure below shows the control/status registers and the read/write data registers. Transmitter control/status registers 0 and 1 (XCSR0 and 1) and receiver control/status registers 0 and 1 (RCSR0 and 1) operate with serial lines 0 and 1 respectively.

Both serial line units have the same bit assignments. There are four registers for each serial line. They are sequential in this order: 0, receiver status; 2, receiver data; 4, transmitter status; and 6, transmitter data. All unused bits are read as zero. The tables below describe the bit assignments for each register.



RCSR
15
8
7
6
5
3
2
0
NOT USED
NOT USED
RECEIVER DONE (READ ONLY)
RECEIVER INTERRUPT ENABLE(READ/WRITE)
RBUF
15
14
13
12
11
8
7
0
NOT USED
ERROR(READ ONLY)
OVERRUN ERROR(READ ONLY)
FRAMING ERROR (READ ONLY)
PARITY ERROR (READ ONLY)
RECEIVE DATA (7.8 BIT DATA IS RIGHT-JUSTIFIED) IF BIT 7 UNUSED = 7
XCSR
15
8
7
6
5
1
0
NOT USED
NOT USED
TRANSMIT READY(READ ONLY)
TRANSMIT INTERRUPT ENABLE(READ/WRITE)
TRANSMIT BREAK (READ/WRITE)
XBUF
15
8
7
0
NOT USED
TRANSMIT DATA(7.8 BIT DATA IS RIGHT-JUSTIFIED)(WRITE ONLY) ON READ = 0

Figure MXV11-A-5
SLU CSR Formats

Bit Assignments for the Receiver Status Register

Bit	Description
Bit 6	Interrupt enable, read/write. A 1 enables receiver interrupts, a 0 disables interrupts. Cleared by Initialize.
Bit 7	Receiver done, read-only. A 1 indicates that the serial interface has received a character. If enabled by bit 6, receiver done will request an interrupt. Receiver done is cleared by reading the receiver data register or by Initialize.

Bit Assignments for the Receiver Data Register

Bit	Description
Bits 0 through 7	Data bits, read-only. Bit 0 is the least significant bit and bit 7 is the most significant. If seven data bits plus parity is selected, bit 7 will always read as a 0.
Bit 12	Parity error, read-only. A 1 indicates that the word being read in bits 0 through 6 has a parity error. Bit 12 will always read 0 when eight data bits and no parity is selected. Cleared when read or by Initialize.
Bit 13	Framing error, read-only. A 1 indicates that a start bit was detected, but there was no corresponding stop bit. A framing error will be generated when a break is received. Cleared when read or by Initialize.
Bit 14	Overrun error, read-only. A 1 indicates that a word in the receiver buffer had not been read when another word was received and placed in the receiver buffer. Cleared when read or by Initialize.
Bit 15	Error, read-only. A 1 indicates that one or more of bits 12, 13, and 14 are 1. Cleared when read or by Initialize.

Bit Assignments for the Transmitter Status Register

Bit	Description
Bit 0	Break, read/write. When set to a 1, bit 0 causes the serial output signal to go to a SPACE condition. A SPACE condition longer than a character time causes a framing error when it is received and is regarded as a break. Cleared by writing a 0 or by Bus Initialize.
Bit 6	Interrupt enable, read/write. A 1 enables transmitter interrupts, a 0 disables interrupts. Cleared by Initialize.
Bit 7	Transmitter ready, read-only. A 1 indicates that the serial interface is ready to accept a character into the transmitter data register. If enabled by bit 6, transmitter ready will request an interrupt. Transmitter ready is cleared when data is written into the transmitter data register. It is set by Initialize.

Bit Assignments for the Transmitter Data Register

Bit	Description
Bits 0 through 7	Data bits, write-only. Bit 0 is the least significant bit and bit 7 is the most significant bit. If seven data bits plus parity is selected, bit 7 will not be transmitted. The transmitter data register will read all 0s.

Interrupt Vector Selection
Two consecutive interrupt vectors (one for receive and one for tranmit) are provided for each of the two serial lines. The interrupt vector format is shown in the figure. Each SLU port can be independently configured to operate in one of two ranges: 000 to 074, or 300 to 376. The table below lists the vector addresses which may be assigned to the serial lines. Note that all vector addresses in the 000 to 074 range, except 060, are reserved vector locations. The jumper selectable bits are 3 through 7. Bits 6 and 7 are wired together.

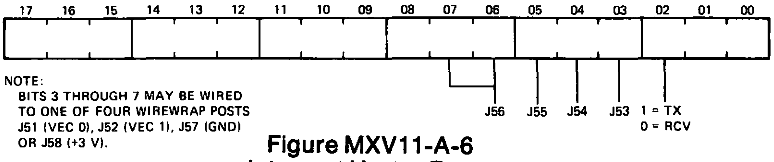

17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
NOTE:
BITS 3 THROUGH 7 MAY BE WIRED TO ONE OF FOUR WIREWRAP POSTS J51 (VEC 0), J52 (VEC 1), J57 (GND) OR J58 (+3 V).
J56
J55
J54
J53
1 = TX
0 = RCV

Figure MXV11-A-6
Interrupt Vector Format

Serial Line Vector Addresses

Serial Line 1 (Console)		Serial line 0 (I/O)
000		300
010		310
020	DIGITAL Reserved	320
030	Do not use	330
040		340
050		350
060	Console	360
070	DIGITAL Reserved	370

The following example illustrates the procedure to configure the vector addresses. Assume that 60 is the address for serial line 1 (console) and 310 is the address for serial line 0 (I/O). The table following describes the relationship between the vector bases, vector address bits, and the jumper posts. The jumpers are configured using the following four rules.

1. If a bit = 1 in both vector bases, it is tied to J58 (Logic 1).
2. If a bit = 0 in both vector bases, it is tied to J57 (Logic 0).
3. If a bit = 1 for serial line 1 and a 0 for serial line 0, it is tied to J52 (VEC 1).
4. If a bit = 0 for serial line 1 and a 1 for serial line 0, it is tied to J51 (VEC 0).

SLU Vector Addresses Example

Serial Line No.	Vector Base	Vector Address Bits 7	6	5	4	3
1 (Console)	60	0	0	1	1	0
0 (I/O)	310	1	1	0	0	1
Jumpers						
From		J56	J56	J55	J54	J53
To		J51	J51	J52	J52	J51

Serial Line Parameter Jumpers

Each MXV11-A serial line has three options that are selected by wirewrap jumpers. The two serial lines may be configured for one or two stop bits, seven data bits plus parity or eight data bits without parity, and odd or even parity.

The parameters are selected by installing jumpers between the appropriate parameter post and J65 (Logic 0) or J66 (Logic 1). The table below describes the jumper configurations required for the desired serial line parameters.

Serial Line Parameter Jumpers

SLU 0 From Pin	To*	SLU 1 From Pin	To*	Function
J62	0	J59	0	7 bits with parity
J62	1	J59	1	**8 bits with no parity
J64	0	J61	0	odd parity
J64	1	J61	1	**even parity
J63	0	J60	0	**1 stop bit
J63	1	J60	1	2 stop bits
J45	J50			**38.4 Kb
		J46	J48	**96 Kb

* Logic 1 = J66
Logic 0 = J65

** Factory Configuration

BAUD RATE JUMPERS

Each serial line can be configured for internal baud rates from 150 to 38.4K baud. Both transmitter and receiver for a given serial line operate at the same baud rate; split baud operation is not provided. One baud rate clock input wirewrap post is provided for each serial line. J46 is the clock input post for serial line 1 and J45 is the clock input post for serial line 0. The baud rate generator outputs are applied to jumper posts J41 through J44 and J47 through J50. The baud rates available at these posts are described in the table below. Configure baud rates (except 110 baud) by connecting a jumper from the desired baud rate generator output post to the serial line clock input post.

The DLV11-KA option may be used to provide 110 baud rate with a 20 mA active or passive current loop interface. The DLV11-KA contains a 110 baud rate clock signal which is supplied to pin 1 of serial lines 0 and 1 I/O connectors. In this case, the MXV11 should be jumpered for an external baud rate.

Baud Rate Jumpers

From	To	Function
J45		SLU0
J46		SLU1
	J41	150 baud clock
	J43	300 baud clock
	J42	1.2K baud clock
	J44	2.4K baud clock
	J47	4.8K baud clock
	J48	9.6K baud clock
	J49	19.2K baud clock
	J50	38.4K baud clock
		External baud clock

Halt/Reboot on Break — A break signal is a continuous spacing condition on the serial data line that occurs either when an operator presses the BREAK key on the associated terminal or when the line is opened. The MXV11-A detects this condition as a framing error. Serial line 1 (console) may be configured for the break responses described in the table below.

Serial Line 1 Break Response Jumpers

Break Response, Operation	**Jumper Posts From**	**To**
Re-Boot	J6	J5
Halt (Factory Configuraton)	J6	J7
No Response	No Jumper installed	

60 Hz Clock — A 60 Hz clock is derived from the crystal on the MXV11. It can be jumpered to the BEVNT line to provide the equivalent of line time clock for a system which does not otherwise have one. In the factory configuration, this signal disconnected. It should not be connected if there is any other source in the system. This includes the case where there is more than one MXV11 module in a system.

This clock can be used with the BDV11 clock status/control register feature. The BDV11 can still be used to turn the clock off under program control, since it accomplishes this by pulling the BEVNT L line to ground on the bus. If this control feature is to be used, the MXV11 should be installed in the same expansion box as the BDV11.

To select this option, jumper J3 to J4 or wire backplane pin AF1 to BR1.

CABLE DEFINITIONS

The following table gives the part numbers, applications and lengths of cabling and options available for the MXV11-A module. DIGITAL offers the BC20M-50 cable for MXV11-A to DLV11-J operation as shown in figure MXV11-A-9. Because longer cables usually require routing without connectors attached, it is recommended the user make cables for lengths greater than 15 meters (50 feet). Cable material must adhere to EIA RS-423 specifications. The connectors on the MXV11-A module are AMP-87272-8 (2 pin × 5 pin on 0.1 inch centers). These connectors can mate with a wide variety of low cost cables including 10-conductor flat cable. Note that pin 1 supplies the SLU clock TTL output when the module's internal clock is selected but is used as the SLU clock input when an external baud rate is desired. Pin 10 carries +12 Vdc used to supply power for the DLV11-KA option. Therefore, pins 1 and 10 should be unterminated if the DLV11-KA option is not used. Cable retention in the module is provided by locking clip contacts (AMP PN87124-1).

The locking clips will hold the receptacle (AMP PN87133-5) in the module connector when the cable is pulled. To remove the cable from the connector, the cable receptacle must be pulled back to disengage the locking clips.

Definition of Cables

Cable	Application	Length
BC21B-05	EIA RS-232C modem cable to interface with modems and acoustic couplers (2 × 5 pin Amp female to RS-232C male)	1.5 m (5 ft)
BC20N-05	EIA RS-232C null modem cable to directly interface with a local EIA RS-232C terminal (2 × 5 pin Amp female to RS-232C female)	1.5 m (5 ft)
BC20M-50	EIA RS-422 or RS-423 cable for high-speed transmission (19.2K baud) (2 × 5 pin Amp female to 2 × 5 Amp female)	15 m (50 ft)
BC05D-10	Extension cable used in conjunction with BC21B-05	3 m (10 ft)
BC05D-25	Extension cable used in conjunction with BC21B-05	7.6 m (25 ft)
BC03M-25	"Null Modem" extension cable used in conjunction with BC21B-05	7.6 m (25 ft)

NOTE

"Strapped" logic levels are provided on data terminal ready (DTR) and request to send (RTS) to all operation of modems with manual provisions (such as Bell 103A data set with 804B auxiliary set).

The MXV11-A may operate with several peripheral device cables and options for flexibility when configuring systems. Figures MXV11-A-8 and -9 show the variety of cables and options used with the MXV11-A as well as the primary applicaton of each.

1. The receivers on the MXV11 have differential inputs. Therefore, when designing an RS-232C or RS-423 cable, RECEIVE DATA (pin 7 on the 2 × 5 pin Amp connector) must be tied to signal

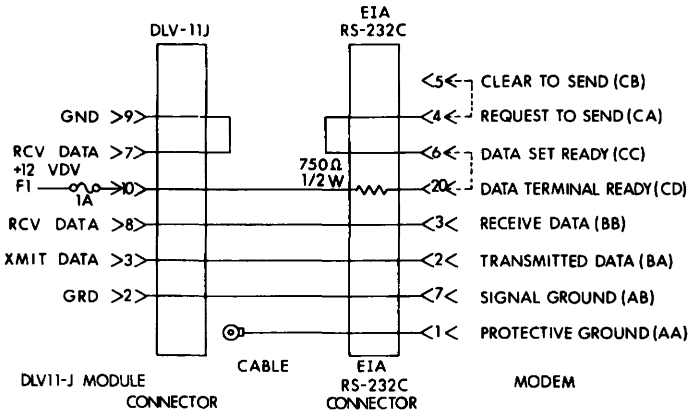
ground (pins 2, 5, or 9) in order to maintain proper EIA levels. (See the figure below.)

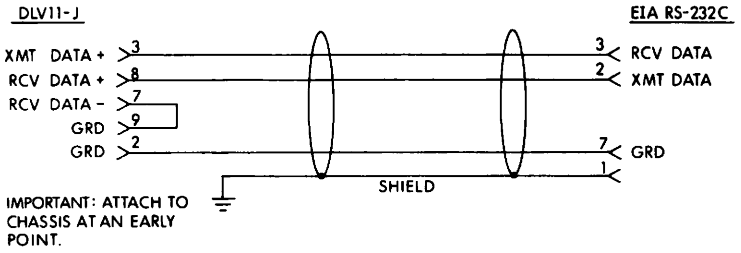
2. To directly connect to a local EIA RS-232C terminal, it is necessary to use a null modem. To design the null modem into the cable, one must switch RECEIVED DATA (pin 2) with TRANSMITTED DATA (pin 3) on the RS-232C male connector as shown in Figure MXV11-A-7.

To mate to the 2 × 5 pin connector block, the following parts are needed:

Cable Receptacle (QTY 1)	AMP PN 87133-5 DEC PN 12-14268-02
Locking Clip Contacts (QTY 9)	AMP PN 87124-1 DEC PN 12-14267-00
Key Pin (pin 6) (QTY 1)	AMP PN 87179-1 DEC PN 12-15418-00

BC21B-05 MODEM CABLE





Interface Connector Pins

Two 10-pin connectors (one for each serial line) are provided on the MXV11-A module. Connector pins and signal functions are described in the following table and shown in the figure below.

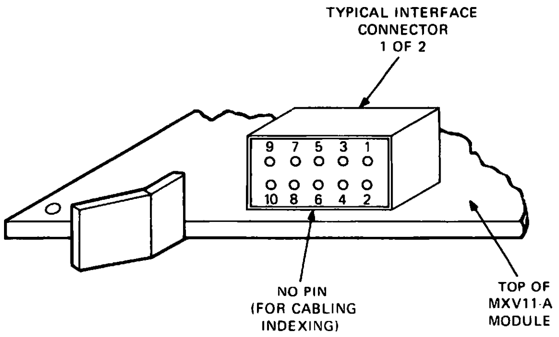

TYPICAL INTERFACE CONNECTOR 1 OF 2
9 7 5 3 1
10 8 6 4 2
NO PIN (FOR CABLING INDEXING)
TOP OF MXV11-A MODULE

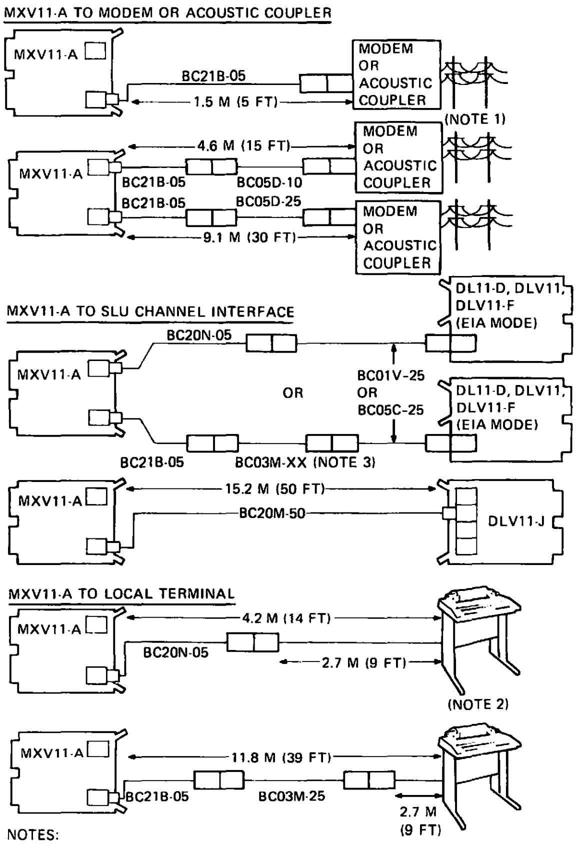
Figure MXV11-A-7
MXV11-A Connector Pins

MXV11-A I/O Connector Pin Functions

Pin	Signal	Function
1	UART CLOCK	The baud rate clock appears on this pin. When an internal baud rate is selected, this pin is a TTL output. When no baud rate is selected on the module, this is an external baud rate input. The high level for the clock >eq 3.0 V.

Pin	Signal	Function
2	Ground	
3	XMIT+	Transmitter output
4	Ground	
5	Ground	
6	NC	Key, pin not provided
7	RCV-	Receiver input most negative
8	RCV+	Receiver input most positive
9	Ground	
10	+12 V	Power for the DLV11-KA option.

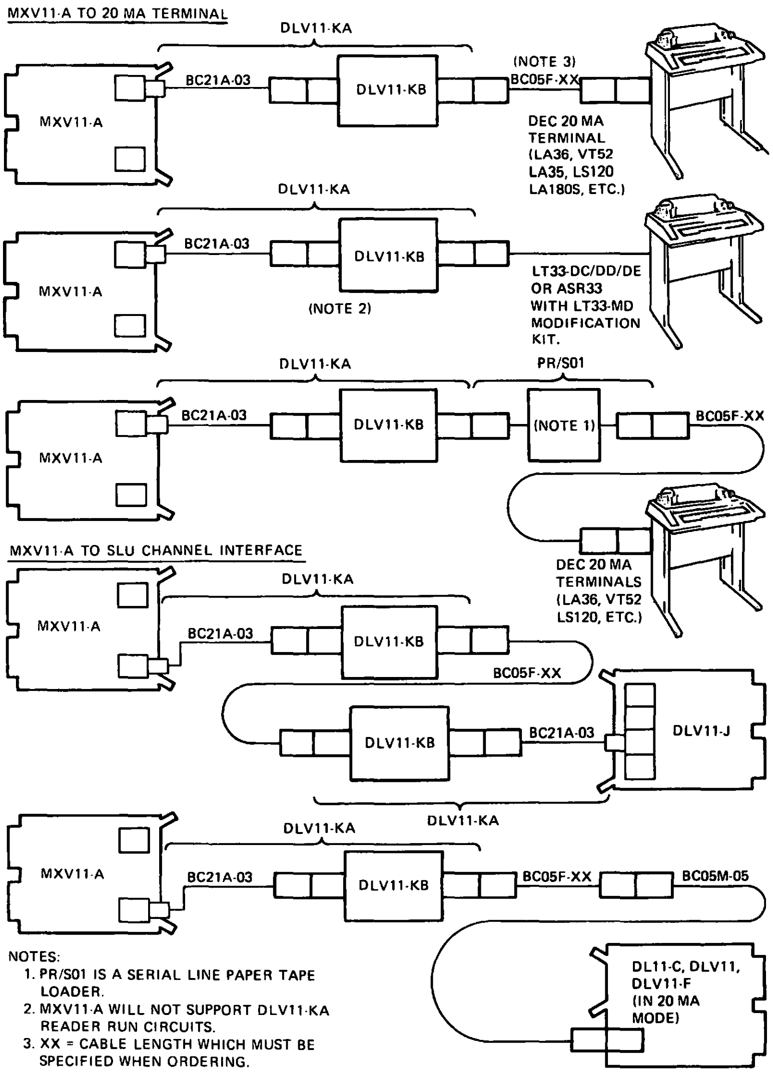
Current Loop
The MXV11-A module can interface with 20 mA active or passive current loop devices when used with the DLV11-KA option. This option consists of a DLV11-KB (EIA to 20 mA current loop converter) and a BC21A-03 interface cable. The MXV11-A does not have the capability to support the reader run portion of the DLV11-KA option. The DLV11-KA option is placed between the MXV11-A serial line output and the 20 mA current loop peripheral device. Figure MXV11-A-9 shows the cables and devices which may be used with the DLV11-KA option.



NOTES:

1. MODEM USED IS A "MANUAL TYPE" SUCH AS BELL 103A WITH 804B.
2. DEC EIS RS-232C TERMINALS (VT52, LA36, LS120, ETC.) COME EQUIPPED WITH A 9 FT CABLE. NON-DEC EIA RS-232C TERMINALS ARE CONNECTED SIMILARLY EXCEPT 9 FT OF LENGTH MUST BE DEDUCTED FROM THE TOTAL CABLE LENGTH.
3. XX = CABLE LENGTH WHICH MUST BE SPECIFIED WHEN ORDERING.

Figure MXV11-A-8
MXV11-A EIA Cable Configurations



NOTES:
1. PR/S01 IS A SERIAL LINE PAPER TAPE LOADER.
2. MXV11-A WILL NOT SUPPORT DLV11-KA READER RUN CIRCUITS.
3. XX = CABLE LENGTH WHICH MUST BE SPECIFIED WHEN ORDERING.

Figure MXV11-A-9
MXV11-A 20 mA Cable Configurations

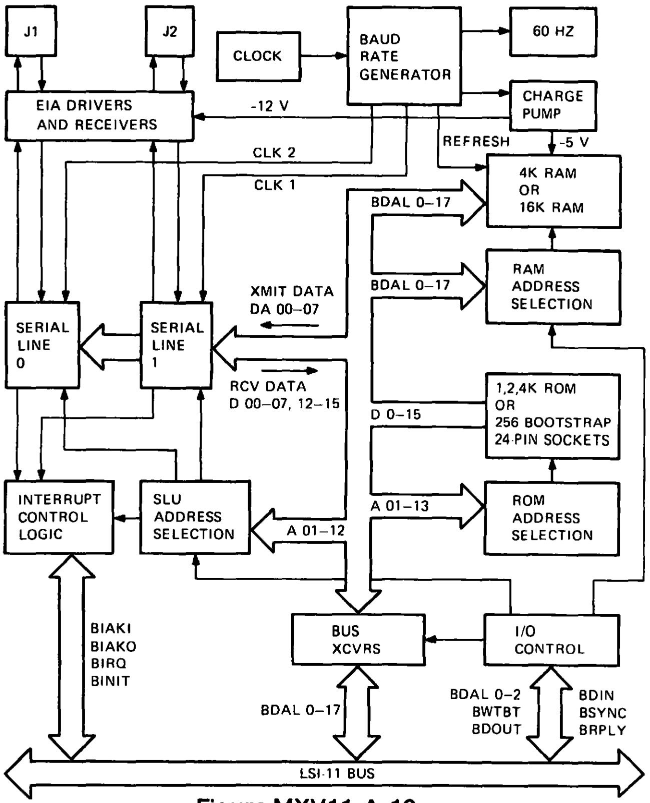


Figure MXV11-A-10
Memory and Serial Line Block Diagram

FUNCTIONAL DESCRIPTION

The console device uses serial line 1 and is connected to J2 whereas an I/O peripheral device or other terminal may use serial line 0 and be connected to J1 on the module. The computer program can address any of four device registers within each channel to transfer data and status information. The computer program can also enable transmitter or receiver interrupts. When a peripheral device requires service, the serial lines will, if enabled, interrupt the program and provide a vector to the necessary service routine.

The 60 Hz signal is derived from the 300 Hz output of the baud rate generator. The 300 Hz signal is applied to a 5 to 1 divider and the resultant 60 Hz output may be used to produce the BEVNT L bus signal. If the current PS bit 7 is clear, BEVNT L will interrupt the processor and go to locations 100 and 102 for a new PC (starting address) and PS (processor status word).

RANDOM ACCESS MEMORY

Memory Array

The memory array contains sixteen 16-pin dynamic RAM integrated circuits (ICs). Eight ICs are used for high byte and eight are used for low byte. Depending on the model, two types of ICs are used. The MXV11-AA uses sixteen 4K × 1 ICs and the MXV11-AC uses 16K × 1 ICs. The figure below shows the signals required to access the memory array. Write byte (WTHB L and WTLB L) control signals are produced by the control logic during a memory write operation to select the addressed byte (DATOB bus cycle); during a word write operation (DATO bus cycle), BWTBT L is high and selects both bytes. Twelve address bits are required for 4K × 1 bit RAMs and fourteen bits for 16K × 1 bit RAMs. The required 12- or 14-bit address is multiplexed over six or seven address lines (MUXA1-A7 L) to all RAM ICs that make up the memory array. Addressing is controlled by row address strobe (RAS L) and column address strobe (CAS L) signals.

Addressing Logic

Addressing is initiated by a master device (either a processor or DMA device) by placing the 16-bit address on the BDAL00-15 L, BAD16 L and BAD17 L lines during the addressing portion of the DATI, DATO (B) or DATIO (B) bus cycles. Bus receivers route the address signals to the address latch where the signals are stored by the assertion of BSYNC L. Address bits A13-17 H are routed to the address decoder which is a 1K × 4 PROM. The SYNC L and BS7 H signals enable the address decoder. If the BBS7 L signals is asserted, memory cannot be accessed, BS7 is high, and the address decoder is inhibited. Three jumper posts (J30, J31 and J32) are connected to J34 (+3 V) and J33 (GND) in straight binary combinations to select the 4K bank starting address. Jumpers W4 and W5 are factory configured and only one is installed, depending on the memory size. When an address is received that resides in the module's user-configured range, RAM SEL L goes low, and enables the outputs of the data latches. RAM SEL L is also sent to the timing and control logic. The address latch also sends address bits A01-A14 H to the address multiplexer to address the memory array with two 6-bit addresses for a 4K memory or two 7-bit addresses for a 16K memory. Jumpers W1 and W3 are factory configured to permit MUXA7 L to become active or to ground it, depending on memory size. Jumper W1 is installed with 4K memory, while W3 is installed when the module contains 16K of memory.

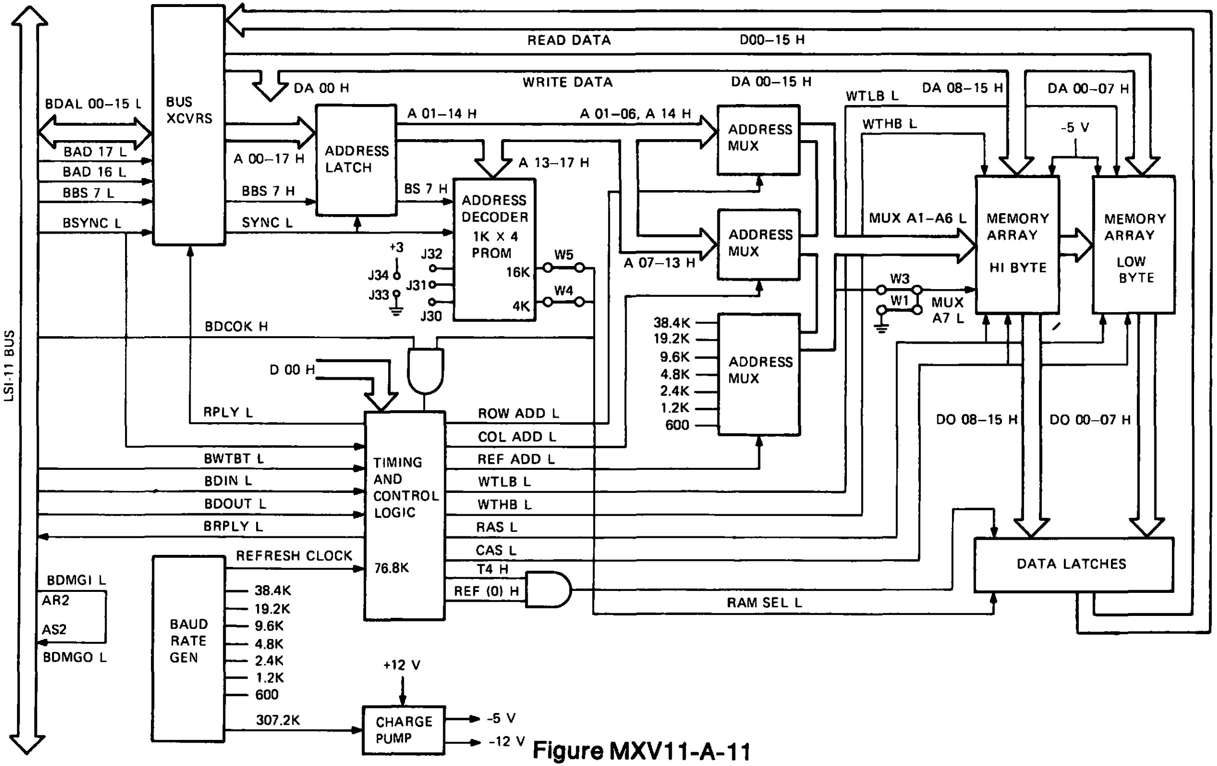


Figure MXV11-A-11
MXV11-A RAM Block Diagram

Timing and Control Logic

Basic timing for any memory cycle is produced by a tapped delay line and appropriate gating logic. Additional logic functions arbitrate refresh cycles, produce control signals for the memory array, addressing logic, and generate the BRPLY signal during any memory access operation. The timing and control logic responds to the RAM L signal by generating the row address strobe (RAS L), row address (ROW ADD L), column address strobe (CAS L), and column address (COL ADD L) signals in the proper timing sequence.

Memory Read

When in a memory read operation, the bus master device asserts BDIN L. The data from the accessed memory location is present on the D00-15 H bus and the bus driver inputs. Reply logic responds to BDIN L by generating an active RPLY L signal to gate the memory read data onto the BDAL00-15 L lines for input to the requesting device. Reply logic also asserts BRPLY L to complete the data transfer portion of the cycle.

Memory Write

When in a memory write operation (or the write portion of a DATIO(B) cycle), the addressing portion is similar to the read cycle addressing. After the addressing portion of the cycle is completed, the master device asserts BDOUT L, and BWTBT L either goes passive (high) if a DATO (word) write cycle is performed, or remains asserted if a DATOB (byte) write cycle is performed. Word/byte select logic responds to the DATO cycle by asserting WTLB L and WTHB L for the duration of the cycle, enabling DA00-15 H to be written into the address location in all memory chips. When in a DATOB cycle, with BDAL00 L asserted, DA00 H (high byte) will cause WTHB L to be active, enabling the writing of DA08-15 H into the eight chips comprising the high (odd) byte of the memory array. Similarly, if BDAL00 L is unasserted, DA00 H will be unasserted and cause WTLB L to become active, enabling the writing of DA00-07 H into the eight chips comprising the low (even) byte of the memory. The reply logic also responds to the active BDOUT L signal by asserting BRLY L, indicating that data has been written, thus completing the data transfer.

Memory Refresh

Dynamic MOS memory integrated circuits in the memory array require periodic refreshing to retain stored data. This is accomplished by forcing a RAS-only operation on each of 64 row addresses in a 4K memory, or 128 row addresses in a 16K memory. Each row address is refreshed once every 0.83 msec for 4K memory and 1.66 msec for 16K memory. On-board refresh circuits eliminate the need for refresh signals on the LSI-11 bus. The accompanying figure shows the signals required for the refresh operation. The refresh clock (76.8K Hz) is obtained from the baud rate generator and applied to the timing and control logic to produce REF ADD L once every 13 microseconds. REF ADD L is applied to the address multiplexer along with seven clock signals (38.4K, 19.2K, 9.6K, 4.8K, 2.4K, 1.2K, and 600 Hz) obtained from the baud rate generator. These signals increment MUXA1-A7 L each time REF ADD L becomes active, thus sequentially refreshing all row addresses in the memory. REF (0) H is inactive during a refresh operation to prevent memory array outputs DO00-15 H being stored in the data latches. In addition, refresh logic will inhibit the assertion of BRPLY L during a refresh operation.

Charge Pump

The charge pump circuit is a dc-to-dc converter which provides the −12 V power for the serial communications driver and −5 V for the MOS memory array integrated circuits. The input power for this circuit is +12 V. A 307.2K Hz signal obtained from the baud rate generator is applied to a rectifier circuit which produces a −12 V output. The −12 V output is applied to a resistor and zener diode to produce the −5 V output. Note that this circuit eliminates the need for backplane power other than the standard +5 V and +12 V.

READ-ONLY MEMORY

Addressing

A master device can address any 16-bit word in the ROM by placing the appropriate address bits on the BDAL01-13 L lines during the addressing portion of the DATI bus cycle. The figure below shows the data/address lines and bus interface signals required for accessing the read-only memory.

BDAL00 L is not used since this address bit functions only as a byte pointer during DATO(B) and the write portion of the DATIO(B) bus cycles. Bus receivers route BAD16-17 H, DA01-15 H and BBS7 H to the address storage latches where the signals are restored by the assertion of BSYNC L. Address bits A01-08 H and A10 H are routed directly to the high byte (E67) and low byte (E57) ROMs.

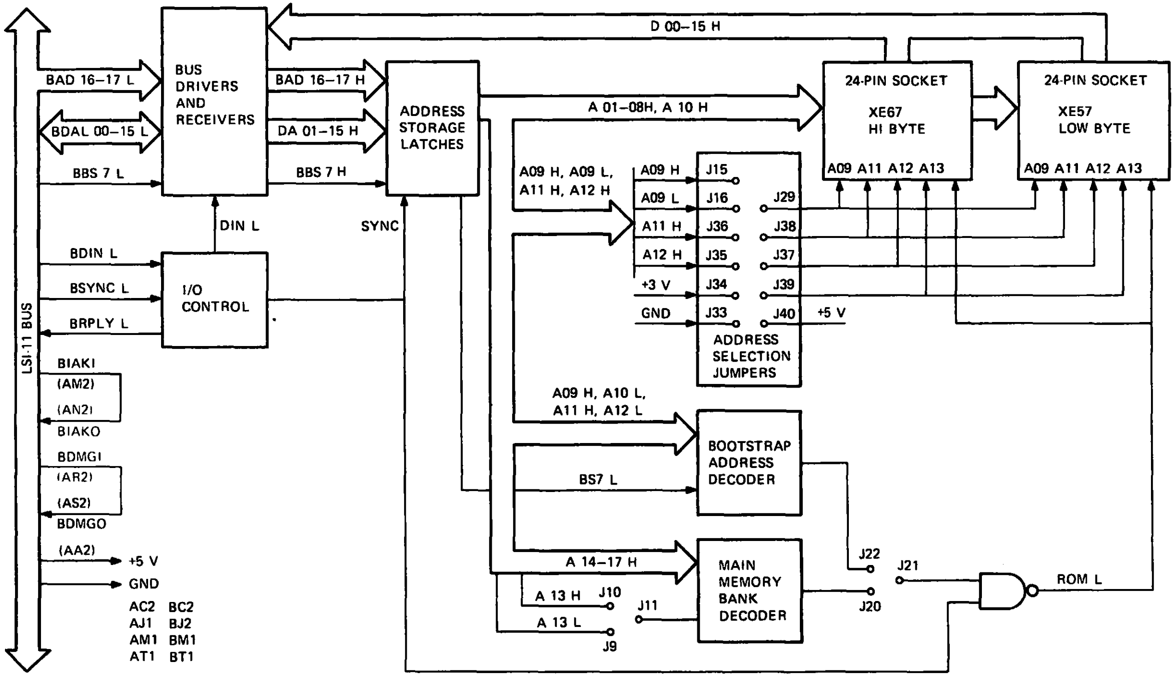


Figure MXV11-A-12
MXV11-A ROM Block Diagram

Bootstrap Address Decoder

Address bits A09 H, A10 L, A11 H, A12 L, and BS7 L are sent to the bootstrap address decoder. When the incoming address is in the bootstrap area of I/O page (173000-173776), a high output is applied to jumper post J22. Jumper post J22 is connected to J21 when the ROM is used to store bootstrap code.

Memory Bank Decoder

Address bits A14-17 H and jumper-selected bits A13 H or A13 L are sent to the main memory bank decoder. Address bit A13 is used to select bank 0 or bank 1. When the incoming address bits are in the selected bank address range, a high output will be applied to jumper post J20. Jumper post J20 is wired to J21 when the ROM is used as main memory.

Address Selection Jumpers

Address bits A09 H, A09 L, A11 H and A12 H are wired to J15, J16, J36 and J35, respectively. These address bits, along with the +3 V (J34), +5 V (J40), and GND (J33) signals, are used to enable and address the 1K, 2K or 4K ROM.

Data Read Operations

Once the ROM chips are addressed, data can be read by the bus master device. BSYNC L is ANDed with the output of the bootstrap address decoder or main memory bank decoder to produce ROM L which enables the ROM outputs. The ROM outputs are sent to the bus drivers on lines D00-15 H. When the processor asserts BDIN L, the bus drivers place the ROM data on BDAL00-15 L. Active BSYNC L and BDIN L also enable the BRPLY bus driver, producing the required response to BDIN L.

I/O Timing and Bus Restrictions

Addressed memory read data is available within 450 ns after the BSYNC L signal is received by the MXV11-A. ROM logic on the module responds only to DATI and DATIO(B) bus cycles. ROM logic functions are not affected by the bus initialize (BINIT L) signal.

SERIAL LINE INTERFACE

General

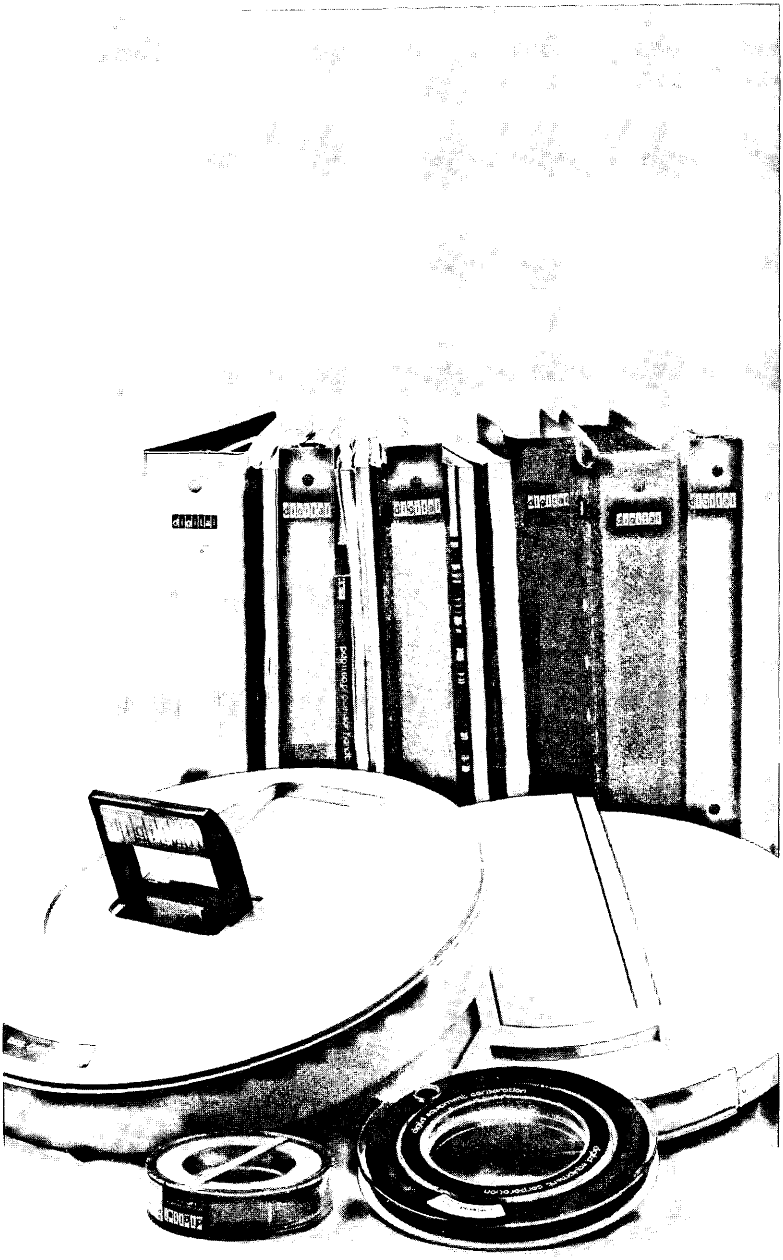
Data passes through four main circuits when moving to and from a peripheral device. The serial transmitter data leaves the SLU and enters the EIA transmitter driver where the data is converted from TTL to EIA-compatible bipolar levels. The data leaves the EIA converter on an interface cable and enters the peripheral device. When using 20 mA current loop signals, an external option (DLV11-KA) is placed between the module output and the 20 mA device. The EIA receiver converts the incoming EIA levels to TTL and applies the data to a receiver buffer in the SLU. The receiver buffer converts the serial data to parallel data and sets a flag. The bus master reads the data. In addition, the SLU monitors the received data and places appropriate error signals on the D12-15 H data bus. The received data and error signals are sent to the output data selector during a read cycle. The output data selector enables the appropriate bus drivers to place the data on the BDAL00-15 L lines.

UART Operation

The serial line units (SLU0 and 1) are universal asynchronous receiver/transmitter (UART) chips. This is a 40-pin LSI chip that is capable of parallel I/O with the computer bus and asynchronous serial I/O with an external device. Jumpers allow the user to select parity functions, number of stop bits and seven or eight data bits. Both transmit and receive functions are totally asynchronous in operation. The transmit and receive clocks are obtained from the same jumper-selectable output of the baud rate generator. Split baud operation is not supported. Clock 1 is used to drive SLU1, while clock 2 drives SLU0. If the internal baud rate generator is not selected, an external baud rate clock may be applied to SLU0 and SLU1 through pin 1 of module connectors J1 and J2.

Baud Rate Generator

The baud rate generator produces the desired UART clock for 150, 300, 1.2K, 2.4K, 4.8K, 9.6K, 19.2K and 38.4K baud rates. A crystal-controlled oscillator produces the basic 19.6608 MHz frequency for the baud rate generator. Two chips in the baud rate generator divide this frequency to produce the available baud rates.


digital

CHAPTER 13
PROMS

This chapter contains specific instructions for programming, loading, and erasing MRV11-AA, MRV11-BA, MRV11-C and MXV11 PROMs. Instructions are also included for using the QJV11 PROM formatter program and the PB11. The QJV11 program reads binary object program paper tapes and produces PROM listings and paper tapes for use with automatic PROM loaders. The *PB11* is a software/hardware option for programming PROMs. It consists of a software utility which runs under RT-11 (V3B or later) and a PROM programmer which interfaces to the computer through a serial line (RS-232C).

PROGRAMMING NOTES

Generally, programs or data that can be read from read/write memory can also be read from MRV11-A PROMs. However, special care is required when using the MTPS instruction and KEV11 option EIS instructions. These instructions are listed below:

Mnemonic	Octal Code	Function
MTPS	1064SS	Move byte to PS
MUL	070RSS	Multiply
DIV	071RSS	Divide
ASH	072RSS	Shift arithmetically
ASHC	073RSS	Arithmetic shift combined

These instructions, when executed, fetch source operands via the *DATIO* bus cycle, rather than the *DATI* bus cycle. Thus, fetching a source operand from a PROM location will result in a bus error (time-out) because the processor will attempt to write into the addressed location after fetching the operand.

There are two ways to avoid this potential problem. The first way involves the MRV11-BA and the MRV11-C UV PROM module because it has the capability to reply to a DATIO bus cycle although the write operation will not actually *write into* PROM. When configured to reply to DATIO cycles, the above list of instructions can be executed from PROM. However, precautions must be taken if a module that is configured to reply to DATIO cycles is used in any system running DIGITAL software or bootstraps. Most DIGITAL software, such as the RT-11 operating system, determines memory size by attempting to write into a location within the memory. If a time-out occurs, it is ensured that no RAM memory is present at that location. PROM memory will normally timeout, as it will not reply to a write cycle. When the MRV11-BA is configured to reply to write cycles, DIGITAL software will assume RAM is present and attempt to write data into it. When the software tries to write data into a PROM, the resultant errors will probably crash the system. Therefore, any MRV11-BA module used with DIGITAL software or bootstraps must *not* be configured to reply to DATIO cycles.

The second way to avoid this potential problem is to include separate MOVe instructions within the program. First, MOVe the source operand from the PROM location to a general register or a location in read/write memory. The MTPS or appropriate EIS instruction is then executed using the general register or the read/write temporary (TEMP) location as the source operand.

Two examples are shown below using general register R4 and memory location TEMP as the source operand.

Using a general register:

```
MOV NEWPS, R4          ;MOVE SOURCE OPERAND FROM PROM
                       ;TO TEMPORARY (GENERAL) REGISTER
MTPS R4                ;MOVE NEWPS TO PS.
```

Using a temporary read/write memory location:

```
MOV CONS,TEMP          ;MOVE SOURCE OPERAND FROM PROM
                       ;TO TEMPORARY LOCATION IN
                       ;READ/WRITE MEMORY.

MUL R1,TEMP    ;MULTIPLY THE CONTENTS OF R1 BY THE
               ;CONSTANT IN TEMP.
```

When programming MRV11-AA, PROMs for use as RT-11 bootstraps, use 256 × 4 PROMs. This will allow the addresses to be configured in the 173000-173776 (773000-773776 for 18-bit processors) range. Processor module power-up mode 2 can then be used for automatically bootstrapping RT-11 during system turn-on. Avoid using 512 × 4 PROMs in this application. If 512 × 4 PROMs are used, the MRV11-AA will respond in the 172000-173776 address range and the RT-11 Editor (EDIT SAV) cannot run properly. This problem exists because the Editor tests for a peripheral device in the 172000-172776 address range. The problem can be avoided by using 256 × 4 PROMs, as described.

LOADING AND INSTALLING PROMS

General

Loading (blasting, burning, or programming) PROMs is the process in which the binary information is stored in the PROM locations. This is a process that must be carefully executed as directed by the appropriate PROM loader manufacturer's instructions.

The procedures for loading and installing PROMs for use in MRV11-AA and MRV11-BA applications are somewhat different.

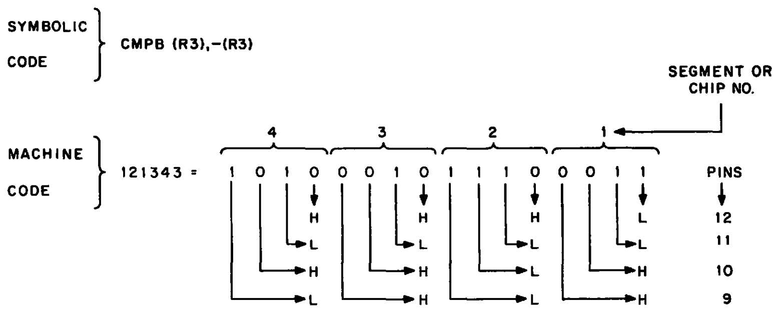
MRV11-AA, MRV11-C, and MXV11 Procedures

PROM Types — Basically, two general types of PROMs can be used in the MRV11-AA module: 512 × 4 bit and 256 × 4 bit. The MRV11-AA module contains sockets for installation of up to 32 PROMs. Only the types listed in this chapter are recommended; the particular pinning and I/O levels for the devices listed are fully compatible with the MRV11-AA addressing and data interface. Note that PROMs are always used in multiples of four, making up the 16-bit word format. Hence, a minimum configuration of four PROMs will comprise either a 256 × 16 or 512 × 16 read-only memory function. Recommended types are listed below.

MRV11-AA PROM Types

Manufacturer or Source	**512 4-Bit PROMs**	**256 4-Bit PROMs**
DIGITAL	MRV11-AC*	—
Intersil	IM5624	IM5623
Signetics	82S131*	82S129
MMI	6306	6301

Word Format — Each PROM word, when read by the processor, is stored in four 4-bit slices in four separate PROMs. Each word is simultaneously addressed and produces its respective 4-bit portion of the 16-bit word that is read. For example, consider the CMPB instruction. Its machine code, using the addressing modes shown, is 121343_8 or 1010001011100011_2. The binary bits are stored in PROMs numbered from 1 to 4. Output pins, as indicated, will yield the read data bits for this instuction when addressed.



MRV11-AA Data Format

Since the word format is contained in four 4-bit slices (one slice in each PROM), the you must load each PROM with successive memory locations. This information can be generated manually—an error-prone, time-consuming process—or it can be generated automatically using the PB11 option or the QJV11 PROM formatter program, described later.

Addressing — PROMs, when installed in the MRV11-AA module, are addressed by low-active address bits. When loading PROMs, you must be careful that the correct addressing technique is used. An example of this addressing technique, relative to PROM pins, is provided in the accompanying table. Note that 256 × 4 bit and 512 × 4 bit PROMs are addressed in exactly the same manner, except for pin 14, which is A8 in the 512 × 4 bit part, and CE in the 256 × 4 bit part. Also note that LSI-11 bus address bit 0 (DAL0 L) is not used in this application since all read operations are 16-bit word bus transfers.

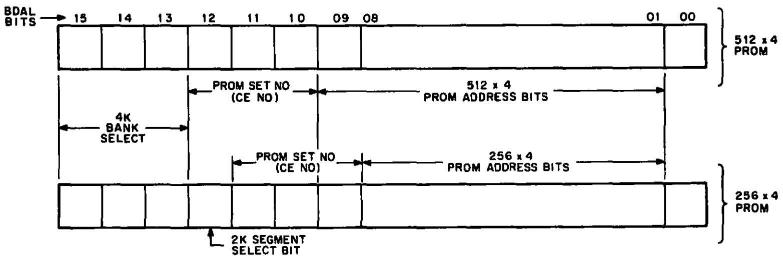
MRV11-AA PROM Addressing

Address*		8	7	6	5	4	3	2	1	←Address (DAL) Bits
Octal	**Binary**	**14**	**1**	**2**	**3**	**4**	**7**	**6**	**5**	**←PROM Chip Pins**
0	000000000	H	H	H	H	H	H	H	H	
2	000000010	H	H	H	H	H	H	H	L	
4	000000100	H	H	H	H	H	H	L	H	
6	000000110	H	H	H	H	H	H	L	L	
10	000001000	H	H	H	H	H	L	H	H	
12	000001010	H	H	H	H	H	L	H	L	Actual
14	000001100	H	H	H	H	H	L	L	H	Logic
16	000001110	H	H	H	H	H	L	L	L	Levels
20	000010000	H	H	H	H	L	H	H	H	Required
o										(256_{10} Locations)
o										
o										
774	111111100	L	L	L	L	L	L	L	H	
776	111111110	L	L	L	L	L	L	L	L	

* Address bit 0 is not used; hence, only even-numbered addresses are shown.

The optional QJV11 PROM formatter program addresses PROMs in the manner described.

The MRV11-AA address format for 512 × 4 and 256 × 4 PROM applications is shown in the accompanying figure. Note that the BDAL0 is not used in the address word format: BDAL1 corresponds to PROM chip address bit A0. The 4K bank select bits and 2K segment select bit (256 × 4 PROM applications only) are jumper-configured on the MRV11-AA module.



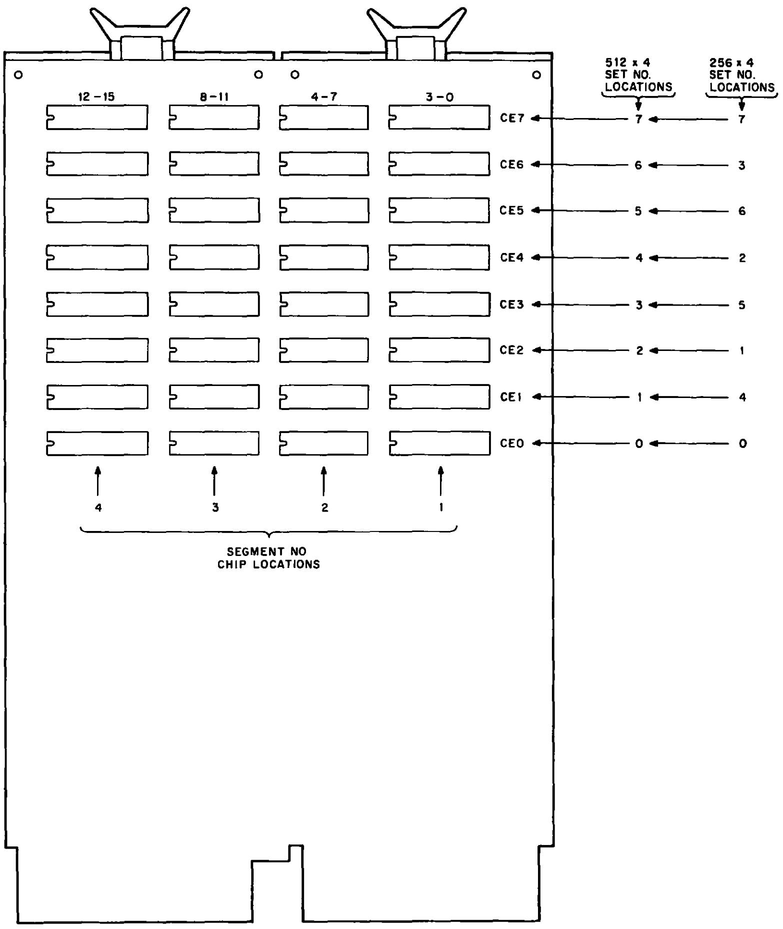
MRV11-AA Address Word Format

Installing PROMS — After PROMs are properly programmed, loaded, and verified, they can be installed on the properly configured MRV11-AA module. Observe that PROMs are always installed in sets of four—one for each segment. Segment and set numbers correspond to those indicated in the QJV11 PROM listing output.

An addressing summary from PROM sets as arranged by physical locations (CE numbers marked on the MRV11-AA module) is provided in the accompanying table.

MRV11-AA PROM Addressing Summary

	512 × 4 PROMs			256 × 4 PROMs		
	Address Range			Address Range		
Set No.	Decimal	Octal	Physical Location	Decimal	Octal	Physical Location
0	0–511	0–11777	CE0	0–255	0–777	CE0
1	512–1023	2000–13777	CE1	256–511	1000–1777	CE4
2	1024–1545	4000–15777	CE2	512–767	2000–2777	CE1
3	1546–2047	6000–17777	CE3	768–1023	3000–3777	CE5
4	2048–2557	10000–11777	CE4	1024–1279	4000–4777	CE2
5	2560–3071	12000–13777	CE5	1280–1545	5000–5777	CE6
6	3072–3583	14000–15777	CE6	1546–1791	6000–6777	CE3
7	3584–4095	16000–17777	CE7	1792–2047	7000–7777	CE7


12-15
8-11
4-7
3-0
CE7
CE6
CE5
CE4
CE3
CE2
CE1
CE0
512 x 4 SET NO. LOCATIONS
256 x 4 SET NO. LOCATIONS
7 7
6 3
5 6
4 2
3 5
2 1
1 4
0 0
4
3
2
1
SEGMENT NO CHIP LOCATIONS

PROM Set and Segment Positions on the MRV11-AA Module

MRV11-BA Procedures

Data Word Format — Each PROM word, when read by the processor, is stored in two bytes in two separate PROMs. Each word is simultaneously addressed and produces its respective 8-bit portion of the 16-bit word that is read. Since the word format is contained in two 8-bit bytes (one byte in each PROM), you must load each PROM with successive memory locations. This information can be generated manually or it can be generated using the optional QJV11 PROM formatter program described later.

Addressing — PROM integrated circuits, when installed in the MRV11-BA module, are addressed by high-active address bits. When loading PROMs, you must be careful that the correct addressing technique is used. An example of this addressing technique, relative to PROM pins, is provided in the accompanying table. Note that LSI-11 bus address bit operations are 16-bit word bus transfers.

Installing PROMs — PROMs should be installed in the MRV11-BA sockets shown in Chapter 12. PROMs are normally installed starting with the first 1K locations (E28 and E29). Check PROM size jumpers to ensure that they agree with the number of PROMs installed. Also, be sure to install the low byte and high byte PROMs in appropriate sockets.

Erasing PROMs — PROMs can be erased by exposure to ultraviolet light at a wavelength of 2537. The recommended integrated light (light intensity × exposure time) is 10 W-s/m^2. The lamp is normally placed approximately 2.54 cm (1 in.) away from the PROM to be erased and turned on for a period of time. The time required can be determined empirically or you can refer to typical times recommended by PROM integrated circuit manufacturers. Typical times may vary from 10 to 30 minutes (approximately).

MRV11-C PROCEDURES

Data Word Format

Each 16-bit word is stored in two bytes. One PROM contains the high byte, the other contains the low byte. The two PROMs are addressed simultaneously. Data is non-inverted and asserted on the high state.

Addressing — MRV11-C uses non-inverted addressing. The address bits are asserted on the high state.

Installing PROMS — PROMs are installed in pairs in their appropriate socket. See Chapter 12, Memories.

Types of PROMs — The MRV11-C can take the following parts:

1K × 8 bit	PROM, UVPROM, OR ROM +5V Parts Only
2K × 8 bit	PROM, UVPROM, OR ROM +5V Parts Only
4K × 8 bit	PROM, UVPROM, OR ROM +5V Parts Only

MXV11-AA, AC

The MXV11-AA and AC have two sockets available for PROM/ROM memory. They are addressed and programmed identically to MRV11-C PROMS. For more information, please refer to Chapter 12.

PB11 UNIVERSAL PROM PROGRAMMER

The PB11 is a complete hardware/software solution to PROM programming requirements for LSI-11 microcomputers. It has been designed for maximum user flexibility, efficiency and ease of operation.

PB11 hardware consists of a table-top PROM programmer unit. The unit is capable of accepting a number of adapter modules which make it possible to program a variety of PROM chips.

It is capable of blasting the following PROMs:

1K × 8 bit	PROM and UVPROM
2K × 8 bit	PROM and UVPROM
4K × 8 bit	UV PROM
256 × 4 bit	PROM
512 × 4 bit	PROM

The software will take the data file and do all the necessary preparation to get the data in the format accepted by the PROM programmer. The data file is then sent to the PROM programmer via the serial link and blasted into the PROMS. No papertape is involved.

The PROM programmer connects to the development system (PDP-11/03 up to PDP-11/34) by means of a serial line. This direct connection to the development system is far simpler to use than an off-line PROM programmer which requires you to create a program and then to physically transfer it to the PROM programmer.

The software utility (which is included in the PB11 package) operates under RT-11 and includes Class C (no support services) software. The entire PROM programming process is controlled from the user's terminal. Easy-to-understand English language commands and diagnostic messages are used to communicate with the operation. PB11 software has been designed to lead the operator through the PROM programming process and to provide diagnostic messages to correct for common errors.

Features

- Convenient desk-top size
- Optional adapters for different PROMs
- Supported under RT-11 development software
- Supports application development under PROMmable FORTRAN
- Includes on-line diagnostic for the PROM programmer
- Connects to serial line interface on host via RS-232C.

Specifications

Physical:

Width:	15.0 in. (38.1 cm)
Depth:	10.75 in. (27.3 cm)

Height:	6.0 in. (15.2 cm)
Weight:	14 lbs. (6.3 Kg)

Environmental

Operating Temperature (Ambient):	5°C to +45°C (41°F to 113°F)
Storage Temperature (Ambient):	−40°C to +5°C (−40°F to 131°F)
Humidity:	to 90% (non condensing)

Electrical:

Input power range:	100, 120, 220, 240 Vac (internally selectable), 50-60 Hz±10% at 35 Watts.

UL listed and CSA certified.

Model Designations

PB11-AY	Desk-top universal PROM programmer with a 25 foot (7.6m) EIA cable for connection to RT-11 (V3V or later) system (11/03 or 11/34) using a serial interface (DLV11-E, DLV11-F, DLV11-J or DL11-W and DL11-E). Includes Class C software on RX01 disk. Adapter kit and RS-232C cable must be purchased with the unit.
PB11-AQ	Same as PB11-AY, includes Class C software on RL01 disk.
PB11K-AA	Adapter for 82S129, 82S131 fusible link PROMs.
PB11K-AB	Adapter for 2708 UV PROMS.
PB11K-AC	Adapter for 82S181, 82S191 fusible link PROMs.
PB11-AD	Adapter for 2716, 2732 UV PROMs.

Documentation provided:

AA-D848A-TC, PROM/RT-11 SYSTEM USERS GUIDE

Hardware Requirements

The PROM programmer comes with a 25 foot (7.6m) EIA cable for connection to a disk-based PDP-11/03 or PDP-11/34 system operating under RT-11 (V3B or later). The user must supply a non-multiplexed serial line interface (DLV11-E, DLV11-F, DLV11-J or DL11-W and DL11-E) with a RS-232C cable and a PROM adapter kit. The PB11 software components are available on RX01 or RL01 media.

Software Support

The PB11 software utility is supported under the RT-11 real-time operating system. Documentation includes a tutorial on PROM-based application design, as well as PROM programmer operation and fault diagnosis.

RT-11 is a powerful foreground/background operating system that supports DIGITAL's special PROMmable FORTRAN—a FORTRAN package uniquely designed for PROM applications.

The PROMmable FORTRAN (RT-11 FORTRAN IV V2.1) is a full implementation of the language, with I/O statements, data formatting and data conversion facilities already provided. Object programs are put out in runtime format, without any intermediate assembly. The FORTRAN package automatically segments user programs into read-only and read/write sections, thus simplifying the work of programming for PROM applications.

All the programming work is done on the development system, then transferred to PROM with minimum programming effort. Because RT-11 is a foreground/background system, users can program PROM in the foreground, while continuing with other software development in the background. The PROM programmer can make programming work more efficient.

The following commands are supported under the software utility:

Command	**Function**
COPY	Copy of an existing PROM chip by reading its contents and replicating in another chip.
DIAGNOSE	Run extended programmer and interface diagnostics to analyze/isolate a hardware problem.
HELP	Print a list of all valid commands on the terminal.
INTERFACE	Alter CSR and vector addresses for serial interface to PROM programmer hardware.
LIST	Print a listing of the contents of a PROM or EPROM chip.
MODIFY	Modify the contents of one or more existing PROM or EPROM chips.
SEQUENTIAL	Redefines PROGRAM, MODIFY, and VERIFY commands to be used to prepare PROMs which were not intended for use with a PDP-11.

VERIFY	Verify that existing PROM or EPROM chips match the contents of a master PROM or program file.

THE QJV11 PROM FORMATTING PROGRAM

The QJV11 PROM formatter program is a paper tape software option that reduces the work required for coding binary patterns for individual PROM chips. Object tapes punched in absolute loader format are the input to the program. QJV11 will produce and verify PROM tapes and listings for PROMs for use in the MRV11-AA and MRV11-BA, and PROMs in other configurations for special user applications.

Loading QJV11

QJV11 is supplied on punched paper tape in absolute loader format. Load the program using the absolute loader program (DEC-11-UABLB-A-PO) or the REV11-A or REV11-C AL (absolute loader) command.

Table 3-4 MRV11-BC PROM Addressing

Address*		10	09	08	07	06	05	04	03	02	01	←Address Bits
Octal	**Binary**	22	23	1	2	3	4	5	6	7	8	←PROM Pins
0	0000000000	L	L	L	L	L	L	L	L	L	L	
2	0000000001	L	L	L	L	L	L	L	L	L	H	
4	0000000010	L	L	L	L	L	L	L	L	H	L	
6	0000000011	L	L	L	L	L	L	L	L	H	H	Actual
10	0000000100	L	L	L	L	L	L	L	H	L	L	Logic
12	0000000101	L	L	L	L	L	L	L	H	L	H	Levels
14	0000000110	L	L	L	L	L	L	L	H	H	L	Required
.												(1024$_{10}$ Locations)
.												
.												
3776	1111111111	H	H	H	H	H	H	H	H	H	H	

*Bus address bit 0 is not used; hence, only even-numbered addresses are shown.

Hardware requirements include 8K read/write memory (minimum), and either a high-speed paper tape reader (CSR address = 177550) or a low-speed paper tape reader (Teletype®) used as the console terminal (CSR address = 177560). (Teletype is a registered trademark of Teletype Corporation.) QJV11 is self-starting; when it has been correctly loaded, the program automatically starts and the initial message, illustrated, is displayed. QJV11 is now ready to receive specific input parameters.

```
PROM V01-00

ENTER AN OCTAL VALUE IN RESPONSE TO QUESTIONS
WHICH REQUIRE A NUMERIC RESPONSE. TYPE 'Y' FOR
YES AND 'N' OR NOTHING FOR NO. TERMINATE ALL
RESPONSES WITH A <CR> (CARRIAGE RETURN).            Initial
RUBOUT MAY BE USED TO DELETE ONE CHARACTER AT       Message
A TIME BEFORE <CR> IS TYPED. CTRL/U MAY BE
USED TO DELETE THE ENTIRE RESPONSE. CTRL/O
MAY BE TYPED TO TURN OFF OUTPUT TO THE
TERMINAL.

HOW MANY WORDS ARE IN A PROM?  1000
HOW MANY BITS ARE IN A PROM WORD?  4
HOW MANY PROMS ARE USED IN PARALLEL?  4
ARE THE DATA BITS INVERTED?  N
ARE THE ADDRESS LINES INVERTED?  Y
HOW MANY BYTES ARE IN THE AREA TO BE                Input
OUTPUT?  20000                                      Parameters
WHAT IS THE STARTING ADDRESS OF THE AREA TO
BE OUTPUT?  0
IS YOUR INPUT/OUTPUT DEVICE ON THE HIGH SPEED
READER/PUNCH?  Y
READY INPUT, TYPE <CR> WHEN READY. <CR>

DO YOU WISH TO PUNCH TAPES?  Y
DO YOU WANT TO VERIFY A TAPE?  N                    QJV11
DO YOU WANT A LIST OF THE PROM CONTENTS?  Y         Operation
DO YOU WANT IT ON A LINE PRINTER?  Y
DO YOU WISH TO MAKE ANOTHER TAPE?  N
```

QJV11 Program Execution (for 512 × 4 PROMs)

Entering Parameters
QJV11 requires certain inputs that must be supplied for each PROM loading session. The dialogue between the QJV11 user and the program is shown in the illustration above for 512 × 4 PROMs (to be used in the MRV11-AA PROM module); a similar example for 1K × 8 PROMs (for use in the MRV11-BA module) is illustrated.

```
PROM V01-00

ENTER AN OCTAL VALUE IN RESPONSE TO QUESTIONS
WHICH REQUIRE A NUMERIC RESPONSE. TYPE 'Y' FOR
YES AND 'N' OR NOTHING FOR NO. TERMINATE ALL
RESPONSES WITH A <CR> (CARRIAGE RETURN).          Initial
RUBOUT MAY BE USED TO DELETE ONE CHARACTER AT     Message
A TIME BEFORE <CR> IS TYPED. CTRL/U MAY BE
USED TO DELETE THE ENTIRE RESPONSE. CTRL/O
MAY BE TYPED TO TURN OFF OUTPUT TO THE
TERMINAL.

HOW MANY WORDS ARE IN A PROM?  2000
HOW MANY BITS ARE IN A PROM WORD?  10
HOW MANY PROMS ARE USED IN PARALLEL?  2
ARE THE DATA BITS INVERTED?  N
ARE THE ADDRESS LINES INVERTED? N
HOW MANY BYTES ARE IN THE AREA TO BE              Input
OUTPUT?  20000                                    Parameters
WHAT IS THE STARTING ADDRESS OF THE AREA TO
BE OUTPUT?  0
IS YOUR INPUT/OUTPUT DEVICE ON THE HIGH SPEED
READER/PUNCH?  Y
READY INPUT, TYPE <CR> WHEN READY.   <CR>

DO YOU WISH TO PUNCH TAPES?  Y
DO YOU WANT TO VERIFY A TAPE?  Y
READY INPUT, TYPE <CR> WHEN READY.   <CR>         QJV11
DO YOU WANT A LIST OF THE PROM CONTENTS?  Y       Operation
DO YOU WANT IT ON A LINE PRINTER?  N
```

QJV11 Program Execution (for 1K × 8 PROMs)

The first parameter to be entered is the number of words (locations) in a PROM. The parameter is requested in the form of a question at the end of the initial message. Operator response to QJV11 requests in the above figures are underlined. Refer to the following table for a list of valid parameter inputs for specific applications.

When reading source tapes for MRV11-AA programs that are *not* greater than 4K, only a single pass of the source tape is required; the QJV11 source buffer is 4K words (4096 × 16 bits). However, longer programs will require one additional pass for each 4K word buffer storage. The appropriate portion of the program is read into the buffer when reading the source tape as specified by the "STARTING ADDRESS OF THE AREA TO BE OUTPUT." Hence, the starting addresses shown in the table are applicable for both multiple-pass programs to specify the starting address for that pass, and for programs that do not reside in the first 4K of system memory (addresses 0-17776).

QJV11 Input Parameters

	MRV11-AA Applications		MRV11-BA Applications	
Parameter	512 × 4 PROMs	256 × 4 PROMs	1K × 8 PROMs	Special Applications
No. words in a PROM (N_8)	1000	400	2000	Any integer power of two (2000 max.)
No. bits in a PROM word (N_8)	4	4	10	1, 2, 4, or 10 (8_{10})
No. PROMs used in parallel	4	4	2	Any number; however, No. bits × No. PROMs must not exceed 20 (16_{10}).
Are data bits inverted	N	N	N	N or Y
Are addr. lines inverted	Y	Y	N	N or Y
How many bytes in the area to be output (N_8)	20000	10000	20000	Any integer power of two (20000 max.)
Starting Address	0, 20000, 40000, 60000, 100000, etc.	0, 10000, 20000, 30000, 40000, etc.	0, 20000, 40000, 60000, 100000, etc.	Any integer multiple of the no. of bytes in the area to be output.
I/O device on the H.S. reader/punch	Y or N	Y or N	Y or N	Y or N

The final input to QJV11 is the source program to be loaded into the PROMs. The program must be in absolute loader format. Place the source tape in the tape reader. Press the RETURN key (shown as <CR> in program examples) on the console device to initiate tape reading.

QJV11 Operation

General — Once the input parameters and source program have been entered, QJV11 is ready to output tapes or listings, or to verify tapes. Operation is simple: respond to QJV11 questions by typing Y or N to indicate the operation(s) desired. The Y answers cause the appropriate QJV11 function to execute immediately. The two examples above do not contain the PROM program listing because the separate line printer was selected for the listing. (QJV11 assumes a line printer CSR address = 177514.) If the line printer was not selected, the listing would appear immediately below the listing request.

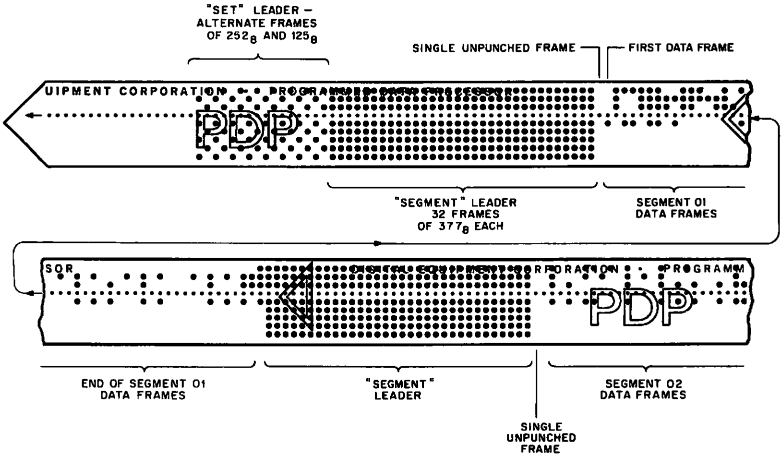
PROM Paper Tape Formats — The QJV11 output tape is punched with as many segments as there are PROMs to be loaded for a particular application. A segment contains the information necessary for loading one PROM. Since multiple PROMs are normally required for the 16-bit PDP-11 word format, either two seqments (MRV11-BA applications) or four segments (MRV11-AA applications) are required, comprising a set. Therefore, the minimum-size QJV11 output would occur when programming a single set of two PROMs for the MRV11-BA or four PROMs for the MRV11-AA.

The tape is punched for MRV11-AA applications as shown below. Special alternate punched frames (16 total) identify that a PROM set follows. This area is followed by 32 frames with all frames punched (377_8), followed by an unpunched frame (0). The first data frame follows immediately after the unpunched frame.

This frame contains the low-order four bits of the 16-bit PROM word at the lowest address (0) in this PROM set; the bits are read over BDAL0-3 bus lines. Successive frames contain 4-bit slices, each representing the 4-bit contents of PROM location. A frame is punched for each of the 256 or 512 locations in the PROM segment. Frames are punched in high-active PROM address sequence, rather than LSI-11 bus address sequence. (LSI-11 bus address bits are inverted; hence, PROMs are programmed starting at the highest bus address or lowest PROM location address.)

The punched tape for MRV11-BA applications will differ from the MRV11-AA type. Instead of the first data frame containing four bits, this frame contains the low-order eight bits of the 16-bit PROM word at the lowest address (0) in this PROM set; the bits are read over BDAL

(0:7) bus lines. Successive frames contain 8-bit bytes, each representing the 8-bit contents of a PROM location. A frame is punched for each of the 1K locations in the PROM segment. Frames for MRV11-BA applications are punched in high-active PROM address sequence.

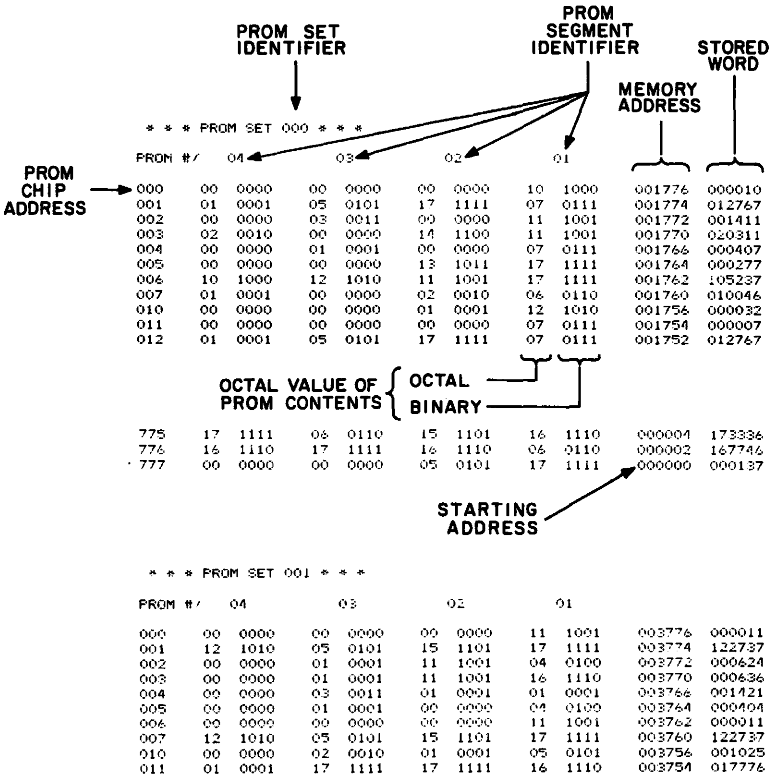

"SET" LEADER – ALTERNATE FRAMES OF 252_8 AND 125_8
SINGLE UNPUNCHED FRAME
FIRST DATA FRAME
UIPMENT CORPORATION • • PROGRAM
PDP
"SEGMENT" LEADER 32 FRAMES OF 377_8 EACH
SEGMENT 01 DATA FRAMES
SOR
RPORATION • • PROGRAM
PDP
END OF SEGMENT 01 DATA FRAMES
"SEGMENT" LEADER
SEGMENT 02 DATA FRAMES
SINGLE UNPUNCHED FRAME

QJV11 PROM Tape Format for MRV11-AC Applications

Verifying Tapes — Tapes punched by QJV11 can be verified by comparing the punched tape with the QJV11 source buffer contents. Respond to the "DO YOU WANT TO VERIFY A TAPE?" request by typing Y <CR>. The program responds with "READY INPUT, TYPE <CR> WHEN READY." Place the tape in the reader and press the RETURN key on the console device.

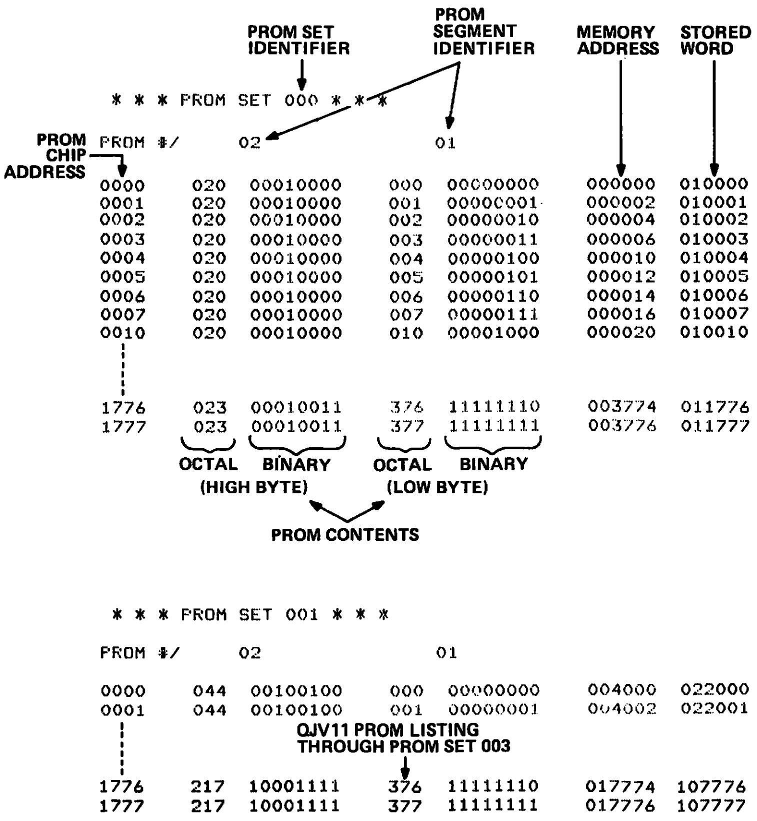
If an error is found, the program responds with "ERROR VERIFYING TAPE." When an error is found, it is necessary to punch another tape. If errors are not found, the program responds with "DO YOU WANT A LIST OF THE PROM CONTENTS?"

QJV11 PROM Listing Formats — Sample portions of PROM listings for MRV11-AA and MRV11-BA applications are shown in the following two figures. The listings are organized by sets. Each set contains the successive PROM addresses, octal and binary codes for each of the PROM sets, the system memory address obtained from the absolute loader format source tape, and the octal content of the 16-bit PROM word.



```
 * * * PROM SET 000 * * *

PROM #/    04            03            02            01

000      00  0000      00  0000      00  0000      10  1000      001776  000010
001      01  0001      05  0101      17  1111      07  0111      001774  012767
002      00  0000      03  0011      00  0000      11  1001      001772  001411
003      02  0010      00  0000      14  1100      11  1001      001770  020311
004      00  0000      01  0001      00  0000      07  0111      001766  000407
005      00  0000      00  0000      13  1011      17  1111      001764  000277
006      10  1000      12  1010      11  1001      17  1111      001762  105237
007      01  0001      00  0000      02  0010      06  0110      001760  010046
010      00  0000      00  0000      01  0001      12  1010      001756  000032
011      00  0000      00  0000      00  0000      07  0111      001754  000007
012      01  0001      05  0101      17  1111      07  0111      001752  012767

775      17  1111      06  0110      15  1101      16  1110      000004  173336
776      16  1110      17  1111      16  1110      06  0110      000002  167746
·777     00  0000      00  0000      05  0101      17  1111      000000  000137

 * * * PROM SET 001 * * *

PROM #/    04            03            02            01

000      00  0000      00  0000      00  0000      11  1001      003776  000011
001      12  1010      05  0101      15  1101      17  1111      003774  122737
002      00  0000      01  0001      11  1001      04  0100      003772  000624
003      00  0000      01  0001      11  1001      16  1110      003770  000636
004      00  0000      03  0011      01  0001      01  0001      003766  001421
005      00  0000      01  0001      00  0000      04  0100      003764  000404
006      00  0000      00  0000      00  0000      11  1001      003762  000011
007      12  1010      05  0101      15  1101      17  1111      003760  122737
010      00  0000      02  0010      01  0001      05  0101      003756  001025
011      01  0001      17  1111      17  1111      16  1110      003754  017776
```

QJV11 PROM Listing Format for MRV11-AA Applications



```
* * * PROM SET 000 * * *

PROM #/        02                   01

0000     020  00010000     000  00000000     000000  010000
0001     020  00010000     001  00000001     000002  010001
0002     020  00010000     002  00000010     000004  010002
0003     020  00010000     003  00000011     000006  010003
0004     020  00010000     004  00000100     000010  010004
0005     020  00010000     005  00000101     000012  010005
0006     020  00010000     006  00000110     000014  010006
0007     020  00010000     007  00000111     000016  010007
0010     020  00010000     010  00001000     000020  010010
 :
 :
1776     023  00010011     376  11111110     003774  011776
1777     023  00010011     377  11111111     003776  011777


* * * PROM SET 001 * * *

PROM #/        02                   01

0000     044  00100100     000  00000000     004000  022000
0001     044  00100100     001  00000001     004002  022001
 :
 :
1776     217  10001111     376  11111110     017774  107776
1777     217  10001111     377  11111111     017776  107777
```

QJV11 PROM Listing Format for MRV11-BA Applications

Using QJV11 PROM Tapes — PROM tapes can be used with automatic PROM loaders, such as those manufactured by DATA I/O Corporation and PRO-LOG Instruments Corporation. Refer to documentation supplied with the PROM loader for the procedure for using the PROM tapes generated by QJV11.

APPENDIX A

TIMING INFORMATION

LSI-11/23 INSTRUCTION TIMING

The following are the assumptions used to calculate instruction times.

Instruction times, are calculated using this equation:

Instruction Time = Basic Time + Source Time + Destination Time

If memory management is enabled, add .16 microseconds for each memory reference. To arrive at incremental value to add to the above instruction time, use the following equation (select numbers from memory cycle column):

Increment = .16 (reads + writes) + .32 (read/modify/write)

All timing is based on the MSV11-D memory (no parity) with the following characteristics. Typical times are shown for a 300 ns microcycle ±10%.

Bus Cycle	Access Time (ns)	Cycle Time (ns)
DATI	210	500
DATO(B)	100	545
DATIO (B)	630	1075

Source Address Time

Instruction	Source Mode	Memory Cycles	Time (Microseconds)
	0	0	0
	1	1	1.12
	2	1	1.12
Double Operand	3	2	2.25
	4	1	1.42
	5	2	2.55
	6	2	2.55
	7	3	3.67

Destination Time

Instruction	Destination Mode	Memory Cycles	Time (Microseconds)
	0	0	0
MOV, CLR, SXT,	1	1	1.84
	2	1	1.84

Instruction	Destination Mode	Memory Cycles	Time (Microseconds)
MFPS, MTPI (D)	3	2	2.66
	4	1	1.84
	5	2	2.96
	6	2	2.96
	7	3	4.09
	0	0	0
CMP, BIT,	1	1	1.42
	2	1	1.42
TST	3	2	2.25
	4	1	1.42
	5	2	2.55
	6	2	2.55
	7	3	3.67
	0	0	0
	1	1	0.22
	2	1	0.22
MTPS, MFPI (D)	3	2	1.05
MUL, DIV,	4	1	1.22
ASH, ASHC	5	2	1.35
	6	2	1.35
	7	3	2.47
	0	0	0
BIC, BIS, ADD,	1	1	2.66
SUB, SWAB,	2	1	2.66
COM, INC,DEC,	3	2	3.49
NEG, ADC, SBC,	4	1	2.66
ROR, ROL, ASR,	5	2	3.79
ASL, XOR	6	2	3.79
	7	3	4.91

Execute and Fetch Time

Instruction	Memory Cycles	Time (Microseconds)
MOV, CMP, BIT, ADD, SUB, BIC, BIC, SXT, CLR,	1	1.72

Instruction	Memory Cycles	Time (Microseconds)
TST, SWAB, COM, INC, DEC, NEG, ADC, SBC, ROR, ROL, ASR, ASL, MFPS		

Instruction	Memory Cycles	Time (Microseconds)
MTPS	1	4.72
MFPI (D)	1	4.12
MTPI (D)	2	2.85
MUL	1	24.52
DIV	1	50.62
ASH	1	30.80
ASHC	1	47.02
All branch instructions	1	1.72
SOB (branch)	1	2.62
(no branch)	1	2.32
RTS	2	3.15
MARK	2	4.65
RTI, RTT	3	5.17
Set or Clear N, Z, V, C	1	2.62
HALT	5	—
WAIT	1	2.92
RESET	1	—
EMT, TRAP	5	7.98
IOT, BPT	5	8.85

JUMP INSTRUCTIONS

Instruction	Mode	Memory Cycles	Time (Microseconds)
	1	1	2.02
	2	1	2.32
JMP	3	2	2.85
	4	1	2.32
	5	2	3.15
	6	2	3.15
	7	3	4.27
	1	2	3.86
	2	2	4.16

Instruction	Mode	Memory Cycles	Time (Microseconds)
JSR	3	3	4.69
	4	2	4.16
	5	3	4.99
	6	3	4.99
	7	4	6.11

Latency

Interrupts (BR requests) are acknowledged at the end of the current instruction. Interrupt latency, which is the time from when an interrupt is requested to when it is granted, is 10.79 microseconds (max) (non-EIS) and 55.65 microseconds (max) (EIS) for the LSI-11/23.

Interrupt service time, which is the time from BR acknowledgement to the first subroutine instruction, is 8.18 microseconds max.

NPR (DMA) latency, which is the time from request to bus mastership for the first NPR device, is 3.34 microseconds max.

FLOATING POINT INSTRUCTION TIMING (OPTION)

The execution time of a floating point instruction is dependent on the following:

1. Type of instruction
2. Type of addressing mode specified
3. Type of memory.

In addition, the execution time of many instructions, such as ADDF, are dependent on the data.

The table below provides the basic instruction times for addressing mode 0 with a microcycle time of 300 ns. The following tables show the additional time required, using the MSV11-D memory with parity enabled, for instructions with other than mode 0. Refer to the notes for the execution time variations for the data-dependent instructions.

Instruction Times

Instr	Micro-cycles	Mode 0 Time (Micro-seconds)	Notes	Modes 1-7
LDF	28	9.15	1,2,19	Use Table 8-2
LDD	36	11.55	1,2,23	
LDCFD	40	12.75	1,4	
LDCDF	55	17.25	1,5	

Instr	Micro-cycles	Mode 0 Time (Micro-seconds)	Notes	Modes 1-7
CMPF	65	20.25	14,15	
CMPD	71	22.05	14,15	
DIVF	301	91.05	1,29,41,43,44	
DIFD	795	239.25	1,30,42,43,44	
ADDF	121	37.05	1,16,17,18,20,25,27,28,41,43,44	
ADDD	139	42.45	1,16,21,22,24,26,27,28,42,43,44	
SUBF	124	37.95	1,16,17,18,20,25,27,28,41,43,44	
SUBD	142	43.35	1,16,21,22,24,26,27,28,42,43,44	
MULF	264	79.95	1,29,31,41,43,44	
MULD	641	193.05	1,30,32,42,43,44	
MODF	682	205.35	1,26,30,32,33,34,35,41,43,44	
MODD	693	208.65	1,26,30,32,33,34,36,42,43,44	
TSTF	28	9.15	1,2,37	
TSTD	32	10.35	1,2,37	

Instruction Times

Instr	Micro-cycles	Mode 0 Time (Micro-seconds)	Notes	Modes 1-7
STF	18	6.15		Use Table 8-3
STD	26	8.55		
STCDF	65	20.25	1,38	
STCFD	48	15.15	11,4	
CLRF	46	11.55		
CLRD	40	12.75		
ABSF	43	13.65	37	Use both Tables 8-2 and 8-3
ABSD	51	16.05	37	
NEGF	42	13.35	1,37	
NEGD	50	15.75	1,37	
LDFPS	11	4.05		Use Table 8-4
LDEXP	38	12.75	1,2,3,37	
LDCIF	60	18.75	6.8	
LDCID	55	17.25	6,8	
LDCLF	60	18.75	6,7,8,9	
LDCLD	55	17.25	6,7,8,9,	

Instr	Micro-cycles	Mode 0 Time (Micro-seconds)	Notes	Modes 1-7
STFPS	16	5.55		Use Table 8-5
STST	17	5.85		
STEXP	34	10.95	1,2	
STCFI	58	18.15	11,12,39	
STCDI	59	18.45	11,12,39	
STCFL	55	17.25	10,11,13,40	
STCDL	56	17.55	10,11,13,40	
CFCC	12	4.35		No Operands
SETF	14	4.95		
SETD	14	4.95		
SETI	14	4.95		
SETL	14	4.95		

Instruction Times

Addressing Mode	Microcycles* Single Precision	Microcycles* Double Precision	Read/Write Memory Cycles Single Precision	Read/Write Memory Cycles Double Precision	Time (Microseconds) Single Precision	Time (Microseconds) Double Precision
1	6,8,11	8,10,13	2/0	4/0	4.81	6.92
2	7,9,11	9,11,14	2/0	4/0	5.11	7.22
2 Immediate	6,8,11	2,4,7	1/0	1/0	4.05	2.85
3	7,9,11	9,11,14	3/0	5/0	5.86	7.97
4	7,9,11	9,11,14	2/0	4/0	5.11	7.22
5	8,10,13	10,12,15	3/0	5/0	6.16	8.27
6	8,10,13	10,12,15	3/0	5/0	6.16	8.27
7	10,12,15	12,14,17	4/0	6/0	7.52	9.63

*Note: The three numbers (of microcycles) in each set represent three different conditions:

1. If the floating point number is positive
2. If the floating point number is negative and non-0
3. If the floating point number is negative 0 with FIUV flag clear.

Instruction Times

	Microcycles		Read/Write Memory Cycles		Time (Microseconds)	
Addressing Mode	**Single Precision**	**Double Precision**	**Single Precision**	**Double Precision**	**Single Precision**	**Double Precision**
1	3	5	0/2	0/4	2.56	4.82
2	6	8	0/2	0/4	3.46	5.72
2 Immediate	−2	−6	0/1	0/1	0.23	−0.97
3	4	6	1/2	1/4	3.61	5.87
4	6	8	0/2	0/4	3.46	5.72
5	5	7	1/2	1/4	3.91	6.17
6	5	7	1/2	1/4	3.91	6.17
7	7	9	2/2	2/4	5.27	7.53

Instruction Times

Instruction	Microcycles		Read/Write Memory Cycles		Time (Microseconds)	
Addressing Mode	Single Integer	Double Integer	Single Integer	Double Integer	Single Integer	Double Integer
1	2	4	1/0	2/0	1.35	2.71
2	3	5	1/0	2/0	1.65	3.01
2						
Immediate	1	1	1/0	1/0	1.05	1.05
3	3	5	2/0	3/0	2.41	3.76
4	3	5	1/0	2/0	1.65	3.01
5	4	6	2/0	3/0	2.71	4.06
6	4	6	2/0	3/0	2.71	4.06
7	6	8	3/0	4/0	4.06	5.42

Instruction Times

Instruction	Microcycles		Read/Write Memory Cycles		Time (Microseconds)	
Addressing Mode	Single Integer	Double Integer	Short	Long	Single Integer	Double Integer
1	2	4	0/1	0/2	1.43	2.86
2	3	5	0/1	0/2	1.73	3.16
2						
Immediate	1	1	0/1	0/1	1.13	1.13
3	3	5	1/1	1/2	2.48	3.91
4	3	5	0/1	0/2	1.73	3.16
5	4	6	1/1	1/2	2.78	4.21
6	4	6	1/1	1/2	2.78	4.21
7	6	8	2/1	2/2	4.13	5.57

NOTES

1. Add 300 ns if result is positive.
2. Add 300 ns if result is non-0.
3. Add 900 ns if SRC > 177 or SRC < −177.
4. Add 900 ns if floating point number = 0.
5. Add 3.3 microseconds if overflow on rounding.
6. Add 300 ns if integer is negative.
7. Add 1.5 microseconds if absolute value of integer < 65,536.

8. Add 1.2 microseconds n times where n = 240 − exp and if absolute value of integer > = 65,536.
9. Add 600 ns if exp < 20.
10. Add 2.1 microseconds n times where n is the smaller of the two absolute values: (310 − exp) or (230 − exp).
11. Add 600 ns if integer is negative.
12. Add 900 ns if integer is negative.
13. Add 1.2 microseconds if floating point numbers are equal.
14. Add 2.1 microseconds if numbers are unequal but the signs are the same.
15. Add 600 ns if FPACC > FPSRC.
16. Add 2.4 microseconds if FPSRC > FPACC.
17. Add 600 ns if adding opposite signs or subtracting like signs.
18. Add 2.4 microseconds if trapped on undefined variable.
19. Add 900 ns and 1.2 microseconds n times where n = exp difference.
20. Add 3.6 microseconds if FPSRC > FPACC.
21. Add 1.2 microseconds if adding opposite signs or subtracting like signs.
22. Add 1.2 microseconds if trapped on undefined variable.
23. Add 900 ns and 1.8 microseconds n times where n = exp difference.
24. Add 1.2 microseconds n times where n = shifts to normalize.
25. Add 1.8 microseconds n times where n = shifts to normalize.
26. Add 3.3 microseconds if underflow.
27. Add 600 ns if overflow.
28. Add 600 ns if need to normalize after multiply or divide.
29. Add 1.2 microseconds if need to normalize after multiply or divide.
30. Add 600 ns for every "1" bit in multiplier (FPSRC).
31. Add 1.2 microseconds for every "1" bit in multiplier (FPSRC).
32. Add 900 ns n times where n = exp module 16 (calculate integer and fraction).
33. Add 300, 600, or 900 ns if exp = 21-40, 41-60 or > 100, or 61-100 respectively.
34. Add 1.8 microseconds if the fractional part = 0.
35. Add 1.2 microseconds if the fractional part = 0.
36. Add 4.5 microseconds if trapped on any of the FP11 interrupts.

37. Add 5.4 microseconds if trapped on overflow.
38. Add 24.3 microseconds if trapped on conversion error.
39. Add 24.9 microseconds if trapped on conversion error.
40. Add 1.2 microseconds if rounding.
41. Add 1.8 microseconds if rounding.
42. Add 8.1 microseconds if trapped on overflow.
43. Add 9.9 microseconds if trapped on underflow.

Interrupt Latency
For all floating point instructions except ADD, SUB, MUL, DIV, and MOD, the interrupt latency is the length of the instruction. The longest execution for each of the FPP instructions are shown in the table below.

In the floating point arithmetic instructions, interrupts may be serviced while in the midst of their execution. The table below shows the longest times between checking for interrupt requests during the floating point instructions. If an interrupt is to be serviced before execution is complete, the instruction is aborted and all the PDP-11 general registers and floating point registers are restored to their original values. After the interrupt is serviced, the floating point instruction is restarted from scratch. This interrupt restore routine takes 6.9 microseconds and the time must be added to the interrupt latency times where execution of an instruction is aborted.

Longest Execution Times

	Worst Case (Mode 7)	**Time**
Instruction	**No. of Microcyles**	**(Microseconds)**
LDF	50	18.77
LDD	56	22.08
LDCFD	56	20.87
LDCDF	91	33.48
CMPF	86	29.57
CMPD	96	34.08

Instruction	Worst Case (Mode 7) No. of Microcyles	Time (Microseconds)
DIVF	350	108.77
DIVD	849	259.98
ADDF	310	96.77
ADDD	596	184.08
SUBF	313	97.67
SUBD	599	184.98
MULF	361	112.07
MULD	919	280.98
MULF	1166	353.57
MODD	1311	398.58
TSTF	58	21.17
TSTD	64	24.48
STF	25	11.42
STD	35	16.08
STCDF	98	34.98
STCFD	54	20.12
CLRF	43	16.82
CLRD	49	20.28
ABSF	72	28.54
ABSD	80	34.11

Instruction	Worst Case (Mode 7) No. of Microcyles	Time (Microseconds)
NEGF	72	28.54
NEGD	80	34.11
LDFPS	17	8.11
LDEXP	64	22.21
LDCIF	127	41.11
LDCID	122	39.61
LDCLF	134	43.97
LDCLD	129	42.47
STFPS	22	9.68
STST	25	11.42
STEXP	42	15.68
STCFI	143	45.98
STCDI	144	46.28
STCFL	145	47.42
STCDL	146	47.72
CFCC	12	4.35
SEIF	14	4.95
SETD	14	4.95
SETI	14	4.95
SETL	14	4.95

Longest Interrupt Request Checking Times

Instruction	From	To	Max. No. of Micro-cycles	Max. Time (Micro-seconds)	Max. Latency Time (Micro-seconds)*
ADDF/SUBF	fetch	end	95	32.27	32.27
ADDF/SUBF	fetch	1st svcpt	83	28.67	35.57
ADDF/SUBF	Ist svcpt	2nd svcpt	42	12.60	19.50
ADDF/SUBF	fetch	3rd svcpt	71	25.07	31.97
ADDF/SUBF	1st svcpt	3rd svcpt	48	14.40	21.30
ADDF/SUBF	2nd svcpt	3rd svcpt	11	3.30	10.20
ADDF/SUBF	3rd svcpt	4th svcpt	6	1.80	8.70
ADDF/SUBF	3rd svcpt	end	80	24.00	24.00
ADDF/SUBF	4th svcpt	end	79	23.70	23.70
ADDD/SUBD	fetch	end	105	36.78	36.78
ADDD/SUBD	fetch	1st svcpt	103	36.18	43.08
ADDD/SUBD	1st svcpt	2nd svcpt	56	16.80	24.70
ADDD/SUBD	fetch	3rd svcpt	83	30.18	37.08
ADDD/SUBD	1st svcpt	3rd svcpt	64	19.20	26.10
ADDD/SUBD	2nd svcpt	3rd svcpt	13	3.90	10.80
ADDD/SUBD	3rd svcpt	4th svcpt	8	2.40	9.30
ADDD/SUBD	3rd svcpt	end	88	26.40	26.40
ADDD/SUBD	4th svcpt	end	87	26.10	26.10

Instruction	From	To	Max. No. of Micro-cycles	Max. Time (Micro-seconds)	Max. Latency Time (Micro-seconds)*
MULF	fetch	end	89	30.47	30.47
MULF	fetch	svcpt	82	28.37	35.27
MULF	svcpt	end	93	27.90	27.90
MULD	fetch	end	101	35.58	35.58
MULD	fetch	svcpt	91	32.58	39.48
MULD	svcpt	end	106	31.80	31.80
DIVF	fetch	end	89	30.47	30.47
DIVF	fetch	svcpt	73	25.67	32.57
DIVF	svcpt	end	96	28.80	28.80
DIVD	fetch	end	101	35.58	35.58
DIVD	fetch	svcpt	85	30.78	37.68
DIVD	svcpt	end	112	33.60	33.60
MODF	fetch	end	93	31.78	31.67
MODF	fetch	1st svcpt	86	29.57	36.47
MODF	1st svcpt	2nd svcpt	29	8.70	15.60

Instruction	From	To	Max. No. of Micro-cycles	Max. Time (Micro-seconds)	Max. Latency Time (Micro-seconds)*
MODF	2nd svcpt	3rd svcpt	51	15.30	22.20
MODF	2nd svcpt	end	125	37.50	37.50
MODF	3rd svcpt	end	86	25.80	25.80
MODD	fetch	end	107	37.38	37.38
MODD	fetch	1st svcpt	91	32.58	39½8
MODD	11st svcpt	2nd svcpt	29	8.70	15.60
MODD	2nd svcpt	3rd svcpt	49	14.70	21.60
MODD	2nd svcpt	end	135	40.50	40.50
MODD	3rd svcpt	end	96	28.80	28.80

*Note: These times include the register restore time.

LSI-11 and LSI-11/23 Bus Timing
Even though the LSI-11 bus is asynchronous, many users did not take advantage of this feature by optimizing their custom interfaces. Some custom interfaces that work with LSI-11 and LSI-11/2 processors will not work with the LSI-11/23 because this module performs bus transactions faster than the earlier modules.

NOTE

Any custom interface that meets the LSI-11 bus specification published in the LSI-11 Processor Handbook will work with either the LSI-11 or LSI-11/23 processor modules. Only modules that do not abide by the specification have a potential compatibility problem.

Note the following while studying the table LSI-11 vs. LSI-11/23 Timing Differences:

1. The LSI-11/23 performs bus transactions faster than the LSI-11.
2. Neither the LSI-11/23 nor the LSI-11 run at the maximum bus speed.

Users should review their custom interface designs to make sure that changing things like address set-up time from 285 ns. to 196 ns. will affect their module operation.

All DIGITAL modules meet the bus timing specification and will work with all processor modules. Also, modules designed with the DCK11 Chipkits will work properly with all processors.

LSI-11 vs. LSI-11/23 BUS TIMING DIFFERENCES

	BUS SPEC[1]	**KD11-F**[2]	**KDF11-A**[3]
BSYNC L—BDIN L	100 ns. min.	190 ns.	130 ns.
BSYNC L—BDOUT L	200 ns. min.	285 ns.	260 ns.
ADDRESS SET-UP TIME	150 ns. min.	285 ns.	196 ns.
ADDRESS HOLD TIME	100 ns. min.	d100 ns.	100 ns.
REPLY TO DIN/DOUT INACTIVE TIME	150 ns. min. 1120 ns. max.	720 ns. min. 285 ns. max.	220 ns. min.

NOTES:

1. Bus specification times are at the bus master inside the bus drivers.
2. LSI-11 with 380 ns. microcycle time.
3. LSI-11/23 with 300 ns. microcycle time.

LSI-11 INSTRUCTION EXECUTION TIME

The execution time for an instruction depends on the instruction itself, the modes of addressing used, and the type of memory referenced. In most cases the instruction execution time is the sum of a Basic Time, a Source Address (SRC) Time, and a Destination Address (DST) Time.

INSTR TIME = Basic Time + SRC Time + DST Time

(BASIC Time = Fetch Time + Decode Time + Execute Time)

Some of the instructions require only some of these times. All timing information is in microseconds, unless otherwise noted. Times are typical; process timing can vary ±20 percent. A 350ns microcycle is assumed.

SOURCE AND DESTINATION TIME

MODE	SRC TIME (Word)	SRC TIME (Byte)	DST TIME (Word)	DST TIME (Byte)
0	0	0	0	0
1	1.40 μs	1.05 μs	2.10 μs	1.75 μs
2	1.40	1.05	2.10	1.75
3	3.50	3.15	4.20	4.20
4	2.10	1.75	2.80	2.45
5	4.20	3.85	4.90	4.90
6	4.20	3.85	4.90	4.55
7	6.30	5.95	6.65	7.00

NOTE FOR MODE 2 and MODE 4 if R6 or R7 used with Byte operation, add 0.35 μs to SRC time and 0.70 μs to DST time.

BASIC TIME

DOPS (Double Operand)	DMO	DM1-7
MOV	3.50 μs	2.45 μs
ADD,XOR,SUB,BIC,BIS	3.50	4.20
CMP,BIT	3.50	3.15
MOVB	3.85	3.85
BICB,BISB	3.85	3.85
CMP,BITB	3.15	2.80

NOTE

DMO = Destination Mode 0
DM1-7 = Destination Modes 1 through 7

SOPS (Single Operand)	DM0	DM1-7
CLR	3.85 μs	4.20 μs
INC,ADC,DEC,SBC	4.20	4.90
COM,NLG	4.20	4.55
ROL,ASL	3.85	4.55
TST	4.20	3.85
ROR	5.25	5.95
ASR	5.60	6.30
CLRB,COMB,NEGB	3.85	4.20
ROLB,ASLB	3.85	4.20
INCB,DECB,SBCB,ADC	3.85	4.55
TSTB	3.85	3.50
RORB	4.20	4.90
ASRB	4.55	5.95
SWAB	4.20	3.85
SXT	5.95	6.65
MFPS (1067DD)	4.90	6.65
MTPS (1064SS)	7.00	7.00 *

* For MTPS use Byte DST time not SRC time.
* Add 0.35 μs to instr. time if Bit 7 of effective OPR = 1

JMP/JSR MODE	DST TIME
1	0.70 μs
2	1.40
3	1.75
4	1.40
5	2.45
6	2.45
7	4.20

INSTRUCTION	BASIC TIMES	
JMP	3.50 μs	
JSR (PC = LINK)	5.25	
JSR (PC ≠ LINK)	8.40	
ALL BRANCHES	3.50	(CONDITION MET OR NOT MET)
SOB (BRANCH)	4.90	
SOB (NO BRANCH)	4.20	
SET CC	3.50	
CLEAR CC	3.50	
NOP	3.50	
RTS	5.25	
MARK	11.55	
RTI	8.75 *	
RTT	8.75 * +	

INSTRUCTION	BASIC TIMES
TRAP,EMT	16.80 * μs
IOT,RPT	18.55 *
WAIT	6.30
HALT	5.60
RESET	5.95 + 10.0 μs for INIT + 90.0 μs
MAINT INST. (00021R)	20.30
RSRVD INST. (00022N)	5.95 (TO GET TO μADDRESS 3000)

* If NEW PS HAS BIT 4 or BIT 7 SET ADD 0.35 μs FOR EACH
+ IF NEW PS HAS BIT 4 (T BIT) SET ADD 2.10 μs

EXTENDED ARITHMETIC (KEV11) INSTRUCTION TIMES

EIS Instruction Times

MODE	SRC TIME
0	0.35 μs
1	2.10
2	2.80
3	3.15
4	2.80
5	3.85
6	3.85
7	5.60

INSTRUCTION	BASIC TIME	
MUL	24.0 to 37.0 μs	If both numbers less than 256 in absolute value
	64.0 μs Worst Case	16 bit multiply
DIV	78.0 μs Worst Case	
ASH (RIGHT)	10.1 + 1.75 per shift	
ASH (LEFT)	10.8 + 2.45 per shift	
ASHC (RIGHT)	10.1 + 2.80 per shift	
ASHC (LEFT)	10.1 + 3.15 per shift	

FIS Instruction Times (us)
INST. TIME = BASIC TIME + SHIFT TIME FOR BINARY POINTS + SHIFT TIME FOR NORMALIZATION

INSTRUCTION	BASIC TIME
FADD	42.1 μs
FSUB	42.4

EXPONENT DIFFERENCE	ALIGN BINARY POINTS
0– 7	2.45 μs per shift
8–15	3.50 + 2.45 per shift over 8
16–23	7.00 + 2.45 per shift over 16

EXPONENT DIFFERENCE	NORMALIZATION
0– 7	2.1 μs per shift
8–15	2.1 + 2.1 per shift over 8
16–23	4.2 + 2.1 per shift over 16

INSTRUCTION BASIC TIME (μs)	
FMUL	74.2 to 80.9 μs if either argument has only 7 bits of precision, i.e., the second word of the 32 bit argument is 0.
	121.1 μs worst case (i.e., arguments have more than 7 bits of precision).
FDIV	151 μs typical 232 μs worst case

DMA (DIRECT MEMORY ACCESS) LATENCY
DMA latency, which is the time from request (BDMRL) to bus mastership for the first DMA device, is 6.45 μs, maximum. This time is the longest processor DATIO cycle which occurs for an ASR instruction with destination modes of 1 through 7. DMA requests are honored during memory refresh by the processor.

INTERRUPT LATENCY (ALL TIMES IN MICROSECONDS)

a. If processor is performing memory refresh (regardless whether KEV11 is present):

Time from interrupt request (BIRQ L) to acknowledgement (BIAK L)	118 μs max
Time from acknowledgement (BIAK L) to fetch of first service routine instruction	+16.5 μs max
Total time from request to first service routine instruction	134.5 μs max

b. If processor is not performing memory refresh (and KEV11 not present):

Time from interrupt request (BIRQ L) to acknowledgement (BIAK L) (Longest instruction is IOT)	18.55 μs max
Time from acknowledgement (BIAK L) to fetch of first service routine instruction	+16.5 μs max
Total time from request to first service routine instruction	35.05 μs max

c. If processor is not performing memory refresh and KEV11 option is present:

Time from interrupt request (BIRQ L) to acknowledgement (BIAK L)	27.6 μs max
Time from acknowledgement (BIAK L) to fetch of first service routine instruction	+16.5 μs max
Total time from request to first service routine instruction	44.1 μs max

NOTE

During all KEV11 instructions (EIS and FIS), device and event interrupt requests are periodically scanned. If present, the instruction is aborted and all processor state information is backed up to the beginning of the instruction. After the interrupt is processed, the KEV11 instruction is re-executed from the beginning. Caution should be observed with the frequency of event interrupts; if the frequency is too high, the KEV11 instruction will never complete. It is suggested a maximum frequency of 3.3 kHz be used on the event input if the KEV11 option is present. Without the KEV11, the maximum frequency should not exceed 20 kHz. Both times allow approximately 50 μs for the interrupt service routine.

APPENDIX B
DIFFERENCES

PDP-11 FAMILY DIFFERENCES

The attached PDP-11 Family Differences table illustrates 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.

Activity	LSI-	PDP-11						
	11	11/23	04	05/10	15/20	34	35/40	45
OPR%R, (R)+ or 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	
OPR%R, (R)+ or 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
OPR%R, @(R)+ or OPR%R, @−(R) using the same register as both register and destination: contents of R are incremented (decremented) by 2 before being used as the source operand.		X			X		X	
OPR%R, @(R)+ or 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
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	
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

Activity	LSI-	PDP-11						
	11	11/23	04	05/10	15/20	34	35/40	45
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
JMP %R or JSR reg, %R traps to 4 (illegal instruction).	X	X	X	X	X	X	X	
JMP %R or JSR reg, %R traps to 10 (illegal instruction).								X
SWAB does *not* change V					X			
SWAB clears V	X	X	X	X		X	X	X
Register addresses (177700—177717) are valid program addresses when used by CPU.				X				
Register addresses (177000—177717) time out when used as a program address by the CPU. Can be addresses under console operation. Note addresses cannot be addressed under console for LSI-11 or LSI-11/23.	X	X			X		X	X
Basic instructions noted in PDP-11 Processor Handbook.	X	X	X	X	X	X	X	

Activity	LSI-	PDP-11						
	11	11/23	04	05/10	15/20	34	35/40	45
SOB, MARK, RTT, SXT instructions.	X	X	X			X	X	X
ASH, ASHC, DIV, MUL.	X	X				X	X	X
XOR instruction.	X	X				X	X	X
The external option KE11-A provides MUL, DIV and SHIFT operation in the same data format.				X	X			
The KE11-E (Expansion Instruction Set) provides the instructions MUL, DIV, ASH, ASHC. These new instructions are 11/45 compatible.							X	
The KE11-F adds unique stack ordered floating point instructions: FADD, FSUB, FMUL, FDIV.							X	
The KEV11 adds EIS/FIS instructions.	X							X
SPL instruction.								
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	

Activity	LSI-	PDP-11						
	11	11/23	04	05/10	15/20	34	35/40	45
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.								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	X		X		
RESET instruction consists of 10 microseconds of INIT followed by a 90 microsecond pause. Power fail not recognized until the instruction is complete.	X	X						
No RTT instruction				X	X			
If RTT sets the T bit, the T bit trap occurs after the instructing following RTT.	X	X	X			X	X	X
If RTI sets T bit, T bit trap is acknowledged after instruction following RTI.				X	X			
If RTI sets T bit, T bit trap is acknowledged immediately following RTI.	X	X	X			X	X	X
If an interrupt occurs during an instruction that has the T bit set, the T bit trap is acknowledged before the interrupt.	X	X	X	X	X	X	X	

Activity	LSI-		PDP-11					
	11	11/23	04	05/10	15/20	34	35/40	45
If an interrupt occurs during an instruction and the T bit is set, the interrupt is acknowledged before T bit trap.								X
T bit trap will sequence out of WAIT instruction		X	X	X	X	X	X	
T bit trap will not sequence out of WAIT instuction. Waits until an interrupt.	X							X
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
Only 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 in LSI-11 or LSI-11/23.	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

Activity	LSI-	PDP-11						
	11	11/23	04	05/10	15/20	34	35/40	45
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
The first instruction in an interrupt service is guaranteed to be executed.					X			
Eight general purpose registers.	X	X	X	X	X	X	X	
Sixteen general purpose registers.								X
PSW address, 177776, not implemented must use new instructions, MTPS (move to PS) and MFPS (move from PS).	X							
PSW address implemented, MTPS and MFPS not implemented.			X	X	X			
PSW address and MTPS and MFPS implemented.		X				X	X	X
Only one interrupt level (BR4) exists.	X							
Four interrupt levels exist.		X	X	X	X	X		
Stack overflow not implemented.	X						X	X
Stack overflow below 400 implemented.		X	X	X	X	X		
Red and yellow zone stack overfow implemented.								

Activity	LSI-	PDP-11						
	11	11/23	04	05/10	15/20	34	35/40	45
Odd address trap not implemented.	X	X					X	X
Odd address trap implemented.			X	X	X	X		
FMUL and FDIV instructions implicity use R6 (one push and pop); hence, R6 must be set up correctly.	X						X	X
FMUL and FDIV instructions do not implicitly use R6.								
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						
Due to their execution time, FIS instructions can abort because of a device interrupt.	X						X	X
EIS instructions do a DATIP and DATO bus sequence when fetching source operand.	X						X	
EIS instructions do a DATI bus sequence when fetching source operand.		X						
MOV instruction does just a DATO bus sequence for the last memory cycle.	X	X				X	X	X

Activity	LSI-		PDP-11					
	11	11/23	04	05/10	15/20	34	35/40	45
MOV instruction does a DATIP and DATO bus sequence for the last memory cycle.			X	X	X		X	X
If PC contains non-existent memory address and a bus error occurs, PC wil have been incremented.	X	X	X	X	X	X		
If PC contains non-existent memory address and bus error occurs, PC will be unchanged.								X
If register contains non-existent memory address in mode 2 and a bus error occurs, register will be incremented.	X	X		X	X		X	
Same as above but register is unchanged.			X			X	X	X
If register contains an odd value in mode 2 and a bus error occurs, register will be incremented.	X	X						
If register contains an odd value in mode 2 and a bus error occurs, register will be unchanged.			X	X	X	X	X	X
Condition codes restored to original values after FIS interrupt abort (EIS doesn't abort on 35/40).								

Activity	LSI-	PDP-11						
	11	11/23	04	05/10	15/20	34	35/40	45
Condition codes that are restored after EIS/FIS interrupt abort are indeterminate.	X						X	
Op codes 075040 through 075377 unconditionally trap to 10 as reserved op codes.		X	X	X	X	X		
If KEV11 option is present, op codes 075040 through 075377 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 existing address, a trap to 10 occurs if user microcode is not present. If no KEV11 option is present, a trap to 10 occurs.	X						X	X
Op codes 210 through 271 trap to 10 as reserved op codes.		X	X	X	X	X		
Op codes 210 through 217 are used as a maintenance instruction	X						X	X
Op codes 075040 through 075777 trap to 10 as reserved op codes.		X	X	X	X	X		

Activity	LSI-	PDP-11						
	11	11/23	04	05/10	15/20	34	35/40	45
Only if KEV11 option is present, op codes 075040 through 075377 can be used as escapes to user microcode. Op codes 075400 through 075777 can also be used as escapes to user microcode and KEV11 option need not be present. If no user microcode exists, a trap to 10 occurs.	X						X	X
Op codes 170000 through 177777 trap to 10 as reserved instructions.			X	X	X			
Op codes 170000 through 177777 are implemented as floating point instructions.		X				X	X	
Op codes 17000 through 177777 can be used as escapes to user microcode. If no user microcode exists, a trap to 10 occurs.	X							
CLR, SXT, MFPS, MTPI and MTPD do just a DATO sequence for the last bus cycle.		X						
CLR and SXT do DATIP-DATO sequence for the last bus cycle.	X		X	X	X	X		
MEM.MGT maintenance mode SR0 bit 8 is implemented.						X	X	X
MEM.MGT maintenance mode SR0 bit 8 is not implemented.		X					X	X

Activity	LSI-	PDP-11						
	11	11/23	04	05/10	15/20	34	35/40	45
PS <15:12>, user mode, user stack pointer, and MTPX and MFPX instructions exist even when MEM.MGT is not configured.		X						
PS <15:12>, user mode, user stack pointer and MTPX and MFPX instructions exist only when MEM.MGT is configured.								X
Current mode PS bits <15:14> of 01 or 10 will cause a MEM.MGT trap upon any memory reference.						X	X	
Current mode PS bits <15:14> set to 01 or 10 will be treated as user mode (11) and not cause a MEM.MGT trap.		X					X	X
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 is user mode will only affect PS <3:0> regardless of whether PS address 177776 is mapped.		X						
MFPS in user mode will cause MEM.MGT trap if PS address 177776 not mapped. If mapped, PS <7:0> are accessed.						X		

Activity	LSI-	PDP-11						
	11	11/23	04	05/10	15/20	34	35/40	45
MFPS in user mode will access PS <7:0> regardless of whether PS address 177776 is mapped.		X						
A HALT instruction in user mode traps to 4.								
A HALT instruction in user mode traps to 10.		X				X		X
If an RTT sets the T bit and the next instruction is an RTI, which clears the T bit, the T bit trap will not be taken.						X	X	
Same as above, but the T bit trap will be taken.	X	X	X				X	X

PRIORITY OF TRAPS AND INTERRUPTS

LSI-11	LSI-11/23	PDP-11/04	PDP-11/05,10
BUSERR Trap	CTLERR Trap	BUSERR Trap	BUSERR Trap
Memory Refresh	MMU Trap	Trap Inst.	Trap Inst.
Trap Inst.	BUSERR Trap	Trace Trap	Trace Trap
Trace Trap	PARERR Trap	STOVF Trap	STOVF Trap
PFAIL Trap	Trap Inst.	PFAIL Trap	PFAIL Trap
Bus Halt Signal	Trace Trap	Device Interupts	Device Interrupts
Event Line	STOVF Trap	Console Halt	Console Halt
Device Interrupts	PFAIL Trap	Wait Loop	Wait Loop
Wait Loop	Device Interrupts		
	Bus Halt Signal		
	Wait Loop		

PDP-11/34	PDP-11/35,40	PDP-11/45
ODDAD Trap	PARERR Trap	Console Halt
MMU Trap	MMU Trap	ODDAD Trap
BUSERR Trap	BUSERR Trap	STOVF Trap
PARERR Trap	STOVF Trap	(Red Zone)
Trap Inst.	(Red Zone)	MMU Trap
Trace Trap	Trap Inst.	BUSERR Trap
STOVF Trap	Trace Trap	PARERR Trap
PFAIL Trap	STOVF Trap	STOVF Trap
Device Interrupts	(Yellow Zone)	(Yellow Zone)
Console Halt	PFAIL Trap	PFAIL Trap
Wait Loop	Console Halt	PIRQ
	Device Interrupts	Device Interrupts
	Wait Loop	Wait Loop
		Trace Loop

BUSERR	= Timeout Error
MMU	= Memory Management Trap
PARERR	= Parity Error
ODDAD	= Odd Address Error
STOVF	= Stack Overflow
PFAIL	= Power Fail
Inst.	= Instructions

KD11-F/KD11-HA/KDF11-AA DETAILED COMPARISON

The KDF11-A module uses five bused spare lines that were reserved for future expansion to implement 18-bit addressing (two lines) and 4-level interrupts (three lines). In addition, the processor uses several of the SSPARE lines for test points or for control functions required during manufacturing testing of the boards. These lines should not cause users any problems unless they have inadvertently bused user signals across the backplane on these pins. For a backplane pin assignment comparison, see the table below.

The KDF11-AA uses the LSI-11 bus closer to its specified limits than either the KD11-F or KD11-HA. These bus timing differences, listed in the table, should have no effect on any user of LSI-11 peripherals or memories since a safety margin still exists between actual times and bus limits.

Backplane Pin Assignment Comparison

Line	Backplane Name	KDF11-AA	KD11-F	KD11-HA
AA1	BSPARE1	BIRQ5L	Reserved*	Reserved*
AB1	BSPARE2	BIRQ6L	Reserved*	Reserved*
BP1	BSPARE6	BIRQ7L	Reserved*	Reserved*
AC1	BAD16	BDAL16L	Reserved*	Reserved*
AD1	BAD17	BDAL17L	Reserved*	Reserved*
AE1	SSPARE1	Single Step	Not Used	STOP L
AF1	SSPARE2	SRUNL	SRUNL	SRUNL
AH1	SSPARE3	SRUNL	Not Used	SRUNL
AK1	MSPAREA	Not Used	Not Used	MTOEL
AL1	MSPAREA	Not Used	Not Used	GND
AM2	BIAKIL	MMU STRH	Not Used	Not Used
AR1	BREFL	Not used†	BREFL	Not Used†
AR2	BDMGIL	UBMAAPL	Not Used	Not Used
BC1	SSPARE4	MMU DAL18H	Not Used	SCLK3H
BD1	SSPARE5	MMU DAL19H	Not Used	SWMIB18H
BE1	SSPARE6	MMU DAL20H	Not Used	SWMIB19H
BF1	SSPARE7	MMU DAL21H	Not Used	SWMIB20H
BH1	SSPARE8	CLK DISL	Not Used	SWMIB21H
BK1	MSPAREB	Not Used	4K RAM BIAS	Not Used
BL1	MSPAREB	Not Used	4K RAM BIAS	Not Used

*Even through these lines are not used on the KD11-F and KD11-HA, they are bused on the backplane and terminated for future bus expansion.

†Not used on the KDF11-AA and KD11-HA but terminated in the inactive state to prevent problems with older memories.

All remaining pins are identical among all three processors.

Comparison of KD11-F, KD11-HA, and KDF11-AA Bus Timing

Interval	Bus Specification (ns)	KD11-F (ns)	KD11-HA (ns)	KDF11-AA (ns)
BSYNC L —BDIN L	100	200	188	144
BSYNC L —BDOUT L	200	300	281	288
BSYNC L —BIAK L	325	600	562	435
Address set-up time on bus	150	300	281	180
Address holding on bus	100	100	100	108
Replay to DIN/DOUT inactive time	200	700+400/−0	675+375/−0	225+72/−0

System Differences—LSI-11 vs. LSI-11/23
Here is a list of system differences between the KDF11-A and the KD11-F. It is intended to point out all possible problems that may arise if a KD11-F is removed from a backplane and a KDF11-A is substituted.

1. **KDF11-A has no boot loader in microcode**—KD11-F has the "L" command. Those users who are downline-loading to KDF11-As will have to change their host software to enter the 14-memory-word bootstrap loader via micro ODT. KDF11-A users whose memory size varies will have to self-size the system via micro ODT or enter a PDP-11 program to do the same thing. The LSI-11's boot loader automatically sizes memory.

   Those users still using paper tape and thus invoking the LSI-11 boot from a terminal will have to enter the 14-word PDP-11 boot by hand from the terminal.
2. **KDF11-A will not perform memory refresh, KD11-F does**—The dual LSI-11 board (KD11-HA) does not perform refresh either. Nevertheless, there will be some users who pull out a KD11-F, insert a KDF11-A, and then must do something else to keep refreshed, such as change over to the newer memories that perform refresh locally (e.g., MSV11-C, MSV11-D).

3. **Event line is on level 6 in KDF11-A, on level 4 in KD11-F**—The KDF11-A has the optional capability to support four interrupt levels, so the real time clock on normal PDP-11 systems is attached to Level 6. In the KD11-F, it is attached to Level 4. Users who have written software and have locked out the event line by setting the priority level to 4 will still see the event line interrupt with the KDF11-A. Users will have to set the priority level to 6 or above to lock out the event line.

   DIGITAL software is unaffected since it takes advantage of the fact that in the KD11-F the other two bits of the priority level are mechanized (read/write) but do not do anything. In many cases, users do not lock out the clock at all because they do not want to miss a tick, and if they have, the change is trivial and should be only in a small number of places in their code.
4. **KDF11-A does not bring four extra microcode bits to module connector as KD11-F does**—The KDF11-A does not have four microcode bits like the KD11-F has. Users who are sensing these four bits for one reason or another (e.g., bus initialize, bus error, etc.) will have to make a change in their system.
5. The KDF11-A pulses SRUN (AF1, AH1) during ODT each time a character is transmitted. The KD11-F and KD11-HA modules do not pulse SRUN during ODT. All three modules do pulse the SRUN line each time an instruction is fetched. Those users who use SRUN for other than driving the RUN lamp should be aware that they will get additional pulses.
6. The KDF11-A supports 18-bit addressing whereas the KD11-F and KD11-HA modules only support 16-bit addressing. When the KDF11-A has the MMU enabled, all modules on the bus must be capable of responding to 18-bit addressing. Memory modules must decode all 18 bits. Modules with DMA capability must address 18 bits instead of only 16 bits. I/O interfaces that use BBS7 instead of decoding address bits 13 and above will work with either 16- or 18-bit addressing.
7. There are some differences in instruction execution between the KD11s and the KDF11-A. See Micro Note #053 for details.

ODT DIFFERENCES—LSI-11 AND LSI-11/23

This micro ODT difference list shows that there are some changes in ODT between the LSI-11 CPUs and the LSI-11/23 CPUs. Notably, the LSI-11/23 does not support the "L" command.

In most cases, if you are using ODT from a console terminal, your program will not be affected. However, the slight differences in response to some commands may impact users who have programmed a host computer to emulate a console terminal to down-load programs to the LSI-11.

Micro ODT Differences: LSI-11 vs. LSI-11/23

LSI-11, LSI-11/2	**LSI-11/23**
All characters that are input are echoed except when in the APT command mode, where no characters are echoed. An echoed line feed (LF) will be followed by a carriage return (CR) only (no second (LF) or padding nulls). This method creates a potential timing problem with a TTY ASR33 which types the next character before the print head has completely returned.	All characters that are input in any command mode except the APT mode are echoed except the octal codes 0, 2, 10, 12, 200, 202, 210 and 212. This suppresses echoing (LF)s, nulls (0), STXs (2), and BSs (10)) because an automatic (CR) and (LF) follow. In the APT command mode, no input characters are echoed.
When an address location is open, another location can be opened without explicitly closing the first location (e.g., 1000/123456 2000/054321).	An address location must be explicitly closed by a (CR) or (LF) command before another is opened or else an error (?) will occur and any open location will automatically be closed without altering it contents.
"↑" will open the previous location.	"↑" is illegal and micro ODT prints ?(CR)(LF)@.
"@" will open a location using indirect addressing.	"@" is illegal and micro ODT prints ?(CR)(LF)@.
"←" will open a location using relative addressing.	"?" is illegal and micro ODT prints ?(CR)(LF)@.
"M" will print the contents of an internal CPU register.	"M" is illegal and micro ODT prints ?(CR)(LF)@.
Rubout (ASCII 177) will delete the last character typed in.	Rubout is illegal and micro ODT prints ?(CR)(LF)@.
"L" is the boot loader command which will load the absolute loader.	"L" is illegal and micro ODT prints ?(CR)(LF)@.

LSI-11, LSI-11/2	LSI-11/23
Control-Shift-S command mode (ASCII 23) accepts 2 bytes forming a 16-bit address and dumps 10 bytes in binary format. The 2 input bytes are not echoed.	Control-Shift-S command mode (ASCII 23) accepts 2 bytes forming an 18-bit address with bits (17:16) always zeroes and dumps 10 bytes in binary format. The 2 input bytes are not echoed.
Up to a 16-bit address and 16-bit data may be entered. Leading zeroes are assumed.	Up to an 18-bit address and 18-bit data may be entered. Leading zeroes are assumed.
Incrementing (LF), the address 177776 results in the address 000000.	Incrementing (LF), the addresses 177776, 377776, 577776 and 777776 result in the addresses 000000, 200000, 400000, and 600000 respectively; i.e., the upper 2 bits of the 18-bit address are not affected. They must be explicitly set.
Incrementing a PDP-11 register from R7 prints out "R8" a the contents of R0.	Incrementing a PDP-11 register from R7 prints out "R0" and the contents of R0.
The I/O page is in the address group 17XXXX.	The I/O page is in the address group 77XXXX where address bits (17:12) must be explicit 1s.
The micro ODT mode can be entered from the following sources: a. A PDP-11 HALT instruction. b. A double bus error. c. An asserted HALT line. d. A power-up option. e. An asserted HALT line caused by a DLV11 framing error. f. A micro ODT bus error. g. A memory refresh bus error. h. An interrupt vector time out.	The micro ODT mode can be entered from the following sources: a. A PDP-11 HALT instruction when in kernel mode; the POKL line is low and the HALT jumper option strap is present. b. An asserted HALT line. c. A power up option. d. An asserted HALT line caused by a DLV11 framing error. e. A micro ODT bus error.

LSI-11, LSI-11/2	LSI-11/23
i. A non-existent micro PC address	
A carriage return (CR) is echoed and followed by just a line feed (LF).	A carriage (CR) return is echoed and followed by another (CR) and line feed (LF).
No “H” command.	“H” causes the LSI-11/23 to execute a microcode routine that, in effect, does nothing.

INTEGRATED CIRCUITS

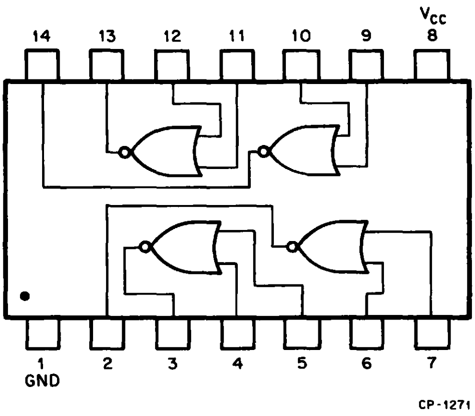


Figure E-1 DEC 8640 QUAD 2-INPUT NOR GATES (Bus Receiver)

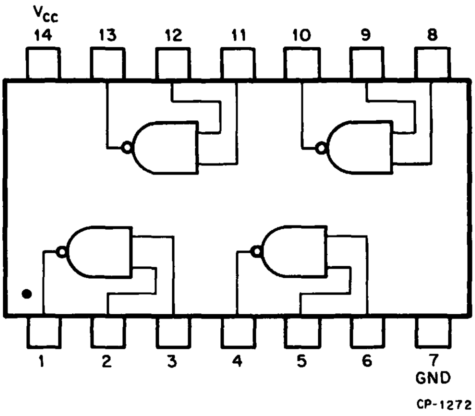


Figure E-2 DEC 8881 QUAD 2-INPUT NAND GATE (Bus Driver)

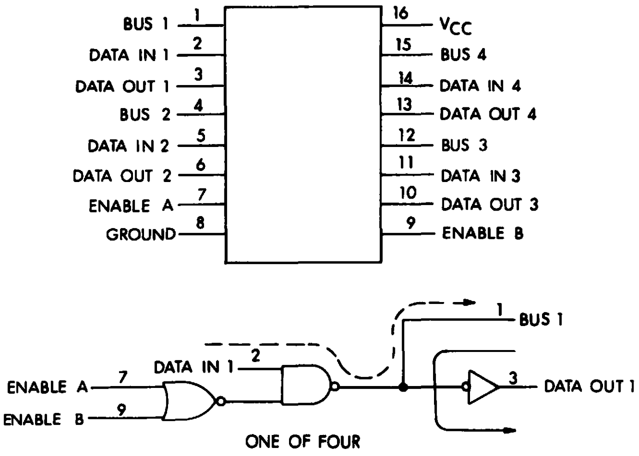


Figure E-3 DEC 8641 QUAD UNIFIED BUS TRANSCEIVER (Bus Receiver/Driver)

APPENDIX D

OCTAL-DECIMAL CONVERSION

OCTAL-DECIMAL INTEGER CONVERSION TABLE

0000 to 0777 (Octal) | 0000 to 0511 (Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal) | 0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

OCTAL-DECIMAL INTEGER CONVERSION TABLE (continued)

2000 to 2777 (Octal) | 1024 to 1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031	2400	1280	1281	1282	1283	1284	1285	1286	1287
2010	1032	1033	1034	1035	1036	1037	1038	1039	2410	1288	1289	1290	1291	1292	1293	1294	1295
2020	1040	1041	1042	1043	1044	1045	1046	1047	2420	1296	1297	1298	1299	1300	1301	1302	1303
2030	1048	1049	1050	1051	1052	1053	1054	1055	2430	1304	1305	1306	1307	1308	1309	1310	1311
2040	1056	1057	1058	1059	1060	1061	1062	1063	2440	1312	1313	1314	1315	1316	1317	1318	1319
2050	1064	1065	1066	1067	1068	1069	1070	1071	2450	1320	1321	1322	1323	1324	1325	1326	1327
2060	1072	1073	1074	1075	1076	1077	1078	1079	2460	1328	1329	1330	1331	1332	1333	1334	1335
2070	1080	1081	1082	1083	1084	1085	1086	1087	2470	1336	1337	1338	1339	1340	1341	1342	1343
2100	1088	1089	1090	1091	1092	1093	1094	1095	2500	1344	1345	1346	1347	1348	1349	1350	1351
2110	1096	1097	1098	1099	1100	1101	1102	1103	2510	1352	1353	1354	1355	1356	1357	1358	1359
2120	1104	1105	1106	1107	1108	1109	1110	1111	2520	1360	1361	1362	1363	1364	1365	1366	1367
2130	1112	1113	1114	1115	1116	1117	1118	1119	2530	1368	1369	1370	1371	1372	1373	1374	1375
2140	1120	1121	1122	1123	1124	1125	1126	1127	2540	1376	1377	1378	1379	1380	1381	1382	1383
2150	1128	1129	1130	1131	1132	1133	1134	1135	2550	1384	1385	1386	1387	1388	1389	1390	1391
2160	1136	1137	1138	1139	1140	1141	1142	1143	2560	1392	1393	1394	1395	1396	1397	1398	1399
2170	1144	1145	1146	1147	1148	1149	1150	1151	2570	1400	1401	1402	1403	1404	1405	1406	1407
2200	1152	1153	1154	1155	1156	1157	1158	1159	2600	1408	1409	1410	1411	1412	1413	1414	1415
2210	1160	1161	1162	1163	1164	1165	1166	1167	2610	1416	1417	1418	1419	1420	1421	1422	1423
2220	1168	1169	1170	1171	1172	1173	1174	1175	2620	1424	1425	1426	1427	1428	1429	1430	1431
2230	1176	1177	1178	1179	1180	1181	1182	1183	2630	1432	1433	1434	1435	1436	1437	1438	1439
2240	1184	1185	1186	1187	1188	1189	1190	1191	2640	1440	1441	1442	1443	1444	1445	1446	1447
2250	1192	1193	1194	1195	1196	1197	1198	1199	2650	1448	1449	1450	1451	1452	1453	1454	1455
2260	1200	1201	1202	1203	1204	1205	1206	1207	2660	1456	1457	1458	1459	1460	1461	1462	1463
2270	1208	1209	1210	1211	1212	1213	1214	1215	2670	1464	1465	1466	1467	1468	1469	1470	1471
2300	1216	1217	1218	1219	1220	1221	1222	1223	2700	1472	1473	1474	1475	1476	1477	1478	1479
2310	1224	1225	1226	1227	1228	1229	1230	1231	2710	1480	1481	1482	1483	1484	1485	1486	1487
2320	1232	1233	1234	1235	1236	1237	1238	1239	2720	1488	1489	1490	1491	1492	1493	1494	1495
2330	1240	1241	1242	1243	1244	1245	1246	1247	2730	1496	1497	1498	1499	1500	1501	1502	1503
2340	1248	1249	1250	1251	1252	1253	1254	1255	2740	1504	1505	1506	1507	1508	1509	1510	1511
2350	1256	1257	1258	1259	1260	1261	1262	1263	2750	1512	1513	1514	1515	1516	1517	1518	1519
2360	1264	1265	1266	1267	1268	1269	1270	1271	2760	1520	1521	1522	1523	1524	1525	1526	1527
2370	1272	1273	1274	1275	1276	1277	1278	1279	2770	1528	1529	1530	1531	1532	1533	1534	1535

3000 to 3777 (Octal) | 1536 to 2047 (Decimal)

	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543	3400	1792	1793	1794	1795	1796	1797	1798	1799
3010	1544	1545	1546	1547	1548	1549	1550	1551	3410	1800	1801	1802	1803	1804	1805	1806	1807
3020	1552	1553	1554	1555	1556	1557	1558	1559	3420	1808	1809	1810	1811	1812	1813	1814	1815
3030	1560	1561	1562	1563	1564	1565	1566	1567	3430	1816	1817	1818	1819	1820	1821	1822	1823
3040	1568	1569	1570	1571	1572	1573	1574	1575	3440	1824	1825	1826	1827	1828	1829	1830	1831
3050	1576	1577	1578	1579	1580	1581	1582	1583	3450	1832	1833	1834	1835	1836	1837	1838	1839
3060	1584	1585	1586	1587	1588	1589	1590	1591	3460	1840	1841	1842	1843	1844	1845	1846	1847
3070	1592	1593	1594	1595	1596	1597	1598	1599	3470	1848	1849	1850	1851	1852	1853	1854	1855
3100	1600	1601	1602	1603	1604	1605	1606	1607	3500	1856	1857	1858	1859	1860	1861	1862	1863
3110	1608	1609	1610	1611	1612	1613	1614	1615	3510	1864	1865	1866	1867	1868	1869	1870	1871
3120	1616	1617	1618	1619	1620	1621	1622	1623	3520	1872	1873	1874	1875	1876	1877	1878	1879
3130	1624	1625	1626	1627	1628	1629	1630	1631	3530	1880	1881	1882	1883	1884	1885	1886	1887
3140	1632	1633	1634	1635	1636	1637	1638	1639	3540	1888	1889	1890	1891	1892	1893	1894	1895
3150	1640	1641	1642	1643	1644	1645	1646	1647	3550	1896	1897	1898	1899	1900	1901	1902	1903
3160	1648	1649	1650	1651	1652	1653	1654	1655	3560	1904	1905	1906	1907	1908	1909	1910	1911
3170	1656	1657	1658	1659	1660	1661	1662	1663	3570	1912	1913	1914	1915	1916	1917	1918	1919
3200	1664	1665	1666	1667	1668	1669	1670	1671	3600	1920	1921	1922	1923	1924	1925	1926	1927
3210	1672	1673	1674	1675	1676	1677	1678	1679	3610	1928	1929	1930	1931	1932	1933	1934	1935
3220	1680	1681	1682	1683	1684	1685	1686	1687	3620	1936	1937	1938	1939	1940	1941	1942	1943
3230	1688	1689	1690	1691	1692	1693	1694	1695	3630	1944	1945	1946	1947	1948	1949	1950	1951
3240	1696	1697	1698	1699	1700	1701	1702	1703	3640	1952	1953	1954	1955	1956	1957	1958	1959
3250	1704	1705	1706	1707	1708	1709	1710	1711	3650	1960	1961	1962	1963	1964	1965	1966	1967
3260	1712	1713	1714	1715	1716	1717	1718	1719	3660	1968	1969	1970	1971	1972	1973	1974	1975
3270	1720	1721	1722	1723	1724	1725	1726	1727	3670	1976	1977	1978	1979	1980	1981	1982	1983
3300	1728	1729	1730	1731	1732	1733	1734	1735	3700	1984	1985	1986	1987	1988	1989	1990	1991
3310	1736	1737	1738	1739	1740	1741	1742	1743	3710	1992	1993	1994	1995	1996	1997	1998	1999
3320	1744	1745	1746	1747	1748	1749	1750	1751	3720	2000	2001	2002	2003	2004	2005	2006	2007
3330	1752	1753	1754	1755	1756	1757	1758	1759	3730	2008	2009	2010	2011	2012	2013	2014	2015
3340	1760	1761	1762	1763	1764	1765	1766	1767	3740	2016	2017	2018	2019	2020	2021	2022	2023
3350	1768	1769	1770	1771	1772	1773	1774	1775	3750	2024	2025	2026	2027	2028	2029	2030	2031
3360	1776	1777	1778	1779	1780	1781	1782	1783	3760	2032	2033	2034	2035	2036	2037	2038	2039
3370	1784	1785	1786	1787	1788	1789	1790	1791	3770	2040	2041	2042	2043	2044	2045	2046	2047

OCTAL-DECIMAL INTEGER CONVERSION TABLE (continued)

4000 to 4777 (Octal)	2048 to 2559 (Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 to 5777 (Octal)	2560 to 3071 (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

OCTAL-DECIMAL INTEGER CONVERSION TABLE (continued)

6000	3072
to	to
6777	3583
(Octal)	(Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079	6400	3328	3329	3330	3331	3332	3333	3334	3335
6010	3080	3081	3082	3083	3084	3085	3086	3087	6410	3336	3337	3338	3339	3340	3341	3342	3343
6020	3088	3089	3090	3091	3092	3093	3094	3095	6420	3344	3345	3346	3347	3348	3349	3350	3351
6030	3096	3097	3098	3099	3100	3101	3102	3103	6430	3352	3353	3354	3355	3356	3357	3358	3359
6040	3104	3105	3106	3107	3108	3109	3110	3111	6440	3360	3361	3362	3363	3364	3365	3366	3367
6050	3112	3113	3114	3115	3116	3117	3118	3119	6450	3368	3369	3370	3371	3372	3373	3374	3375
6060	3120	3121	3122	3123	3124	3125	3126	3127	6460	3376	3377	3378	3379	3380	3381	3382	3383
6070	3128	3129	3130	3131	3132	3133	3134	3135	6470	3384	3385	3386	3387	3388	3389	3390	3391
6100	3136	3137	3138	3139	3140	3141	3142	3143	6500	3392	3393	3394	3395	3396	3397	3398	3399
6110	3144	3145	3146	3147	3148	3149	3150	3151	6510	3400	3401	3402	3403	3404	3405	3406	3407
6120	3152	3153	3154	3155	3156	3157	3158	3159	6520	3408	3409	3410	3411	3412	3413	3414	3415
6130	3160	3161	3162	3163	3164	3165	3166	3167	6530	3416	3417	3418	3419	3420	3421	3422	3423
6140	3168	3169	3170	3171	3172	3173	3174	3175	6540	3424	3425	3426	3427	3428	3429	3430	3431
6150	3176	3177	3178	3179	3180	3181	3182	3183	6550	3432	3433	3434	3435	3436	3437	3438	3439
6160	3184	3185	3186	3187	3188	3189	3190	3191	6560	3440	3441	3442	3443	3444	3445	3446	3447
6170	3192	3193	3194	3195	3196	3197	3198	3199	6570	3448	3449	3450	3451	3452	3453	3454	3455
6200	3200	3201	3202	3203	3204	3205	3206	3207	6600	3456	3457	3458	3459	3460	3461	3462	3463
6210	3208	3209	3210	3211	3212	3213	3214	3215	6610	3464	3465	3466	3467	3468	3469	3470	3471
6220	3216	3217	3218	3219	3220	3221	3222	3223	6620	3472	3473	3474	3475	3476	3477	3478	3479
6230	3224	3225	3226	3227	3228	3229	3230	3231	6630	3480	3481	3482	3483	3484	3485	3486	3487
6240	3232	3233	3234	3235	3236	3237	3238	3239	6640	3488	3489	3490	3491	3492	3493	3494	3495
6250	3240	3241	3242	3243	3244	3245	3246	3247	6650	3496	3497	3498	3499	3500	3501	3502	3503
6260	3248	3249	3250	3251	3252	3253	3254	3255	6660	3504	3505	3506	3507	3508	3509	3510	3511
6270	3256	3257	3258	3259	3260	3261	3262	3263	6670	3512	3513	3514	3515	3516	3517	3518	3519
6300	3264	3265	3266	3267	3268	3269	3270	3271	6700	3520	3521	3522	3523	3524	3525	3526	3527
6310	3272	3273	3274	3275	3276	3277	3278	3279	6710	3528	3529	3530	3531	3532	3533	3534	3535
6320	3280	3281	3282	3283	3284	3285	3286	3287	6720	3536	3537	3538	3539	3540	3541	3542	3543
6330	3288	3289	3290	3291	3292	3293	3294	3295	6730	3544	3545	3546	3547	3548	3549	3550	3551
6340	3296	3297	3298	3299	3300	3301	3302	3303	6740	3552	3553	3554	3555	3556	3557	3558	3559
6350	3304	3305	3306	3307	3308	3309	3310	3311	6750	3560	3561	3562	3563	3564	3565	3566	3567
6360	3312	3313	3314	3315	3316	3317	3318	3319	6760	3568	3569	3570	3571	3572	3573	3574	3575
6370	3320	3321	3322	3323	3324	3325	3326	3327	6770	3576	3577	3578	3579	3580	3581	3582	3583

7000	3584
to	to
7777	4095
(Octal)	(Decimal)

	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591	7400	3840	3841	3842	3843	3844	3845	3846	3847
7010	3592	3593	3594	3595	3596	3597	3598	3599	7410	3848	3849	3850	3851	3852	3853	3854	3855
7020	3600	3601	3602	3603	3604	3605	3606	3607	7420	3856	3857	3858	3859	3860	3861	3862	3863
7030	3608	3609	3610	3611	3612	3613	3614	3615	7430	3864	3865	3866	3867	3868	3869	3870	3871
7040	3616	3617	3618	3619	3620	3621	3622	3623	7440	3872	3873	3874	3875	3876	3877	3878	3879
7050	3624	3625	3626	3627	3628	3629	3630	3631	7450	3880	3881	3882	3883	3884	3885	3886	3887
7060	3632	3633	3634	3635	3636	3637	3638	3639	7460	3888	3889	3890	3891	3892	3893	3894	3895
7070	3640	3641	3642	3643	3644	3645	3646	3647	7470	3896	3897	3898	3899	3900	3901	3902	3903
7100	3648	3649	3650	3651	3652	3653	3654	3655	7500	3904	3905	3906	3907	3908	3909	3910	3911
7110	3656	3657	3658	3659	3660	3661	3662	3663	7510	3912	3913	3914	3915	3916	3917	3918	3919
7120	3664	3665	3666	3667	3668	3669	3670	3671	7520	3920	3921	3922	3923	3924	3925	3926	3927
7130	3672	3673	3674	3675	3676	3677	3678	3679	7530	3928	3929	3930	3931	3932	3933	3934	3935
7140	3680	3681	3682	3683	3684	3685	3686	3687	7540	3936	3937	3938	3939	3940	3941	3942	3943
7150	3688	3689	3690	3691	3692	3693	3694	3695	7550	3944	3945	3946	3947	3948	3949	3950	3951
7160	3696	3697	3698	3699	3700	3701	3702	3703	7560	3952	3953	3954	3955	3956	3957	3958	3959
7170	3704	3705	3706	3707	3708	3709	3710	3711	7570	3960	3961	3962	3963	3964	3965	3966	3967
7200	3712	3713	3714	3715	3716	3717	3718	3719	7600	3968	3969	3970	3971	3972	3973	3974	3975
7210	3720	3721	3722	3723	3724	3725	3726	3727	7610	3976	3977	3978	3979	3980	3981	3982	3983
7220	3728	3729	3730	3731	3732	3733	3734	3735	7620	3984	3985	3986	3987	3988	3989	3990	3991
7230	3736	3737	3738	3739	3740	3741	3742	3743	7630	3992	3993	3994	3995	3996	3997	3998	3999
7240	3744	3745	3746	3747	3748	3749	3750	3751	7640	4000	4001	4002	4003	4004	4005	4006	4007
7250	3752	3753	3754	3755	3756	3757	3758	3759	7650	4008	4009	4010	4011	4012	4013	4014	4015
7260	3760	3761	3762	3763	3764	3765	3766	3767	7660	4016	4017	4018	4019	4020	4021	1022	4023
7270	3768	3769	3770	3771	3772	3773	3774	3775	7670	4024	4025	4026	4027	4028	4029	4030	4031
7300	3776	3777	3778	3779	3780	3781	3782	3783	7700	4032	4033	4034	4035	4036	4037	4038	4039
7310	3784	3785	3786	3787	3788	3789	3790	3791	7710	4040	4041	4042	4043	4044	4045	4046	4047
7320	3792	3793	3794	3795	3796	3797	3798	3799	7720	4048	4049	4050	4051	4052	4053	4054	4055
7330	3800	3801	3802	3803	3804	3805	3806	3807	7730	4056	4057	4058	4059	4060	4061	4062	4063
7340	3808	3809	3810	3811	3812	3813	3814	3815	7740	4064	4065	4066	4067	4068	4069	4070	4071
7350	3816	3817	3818	3819	3820	3821	3822	3823	7750	4072	4073	4074	4075	4076	4077	4078	4079
7360	3824	3825	3826	3827	3828	3829	3830	3831	7760	4080	4081	4082	4083	4084	4085	4086	4087
7370	3832	3833	3834	3835	3836	3837	3838	3839	7770	4088	4089	4090	4091	4092	4093	4094	4095

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

digital microcomputer processor handbook 1979-80

digital

DIGITAL EQUIPMENT CORPORATION, Corporate Headquarters: Maynard, Massachusetts 01754, Telephone (617) 897-5111—SALES AND SERVICE OFFICES; UNITED STATES—ALABAMA, Birmingham, Huntsville • ARIZONA, Phoenix, Tucson • CALIFORNIA, El Segundo, Oakland, Sacramento, San Diego, San Francisco, Santa Ana, Santa Barbara, Santa Clara • COLORADO, Colorado Springs, Denver • CONNECTICUT, Fairfield, Meriden • FLORIDA, Miami, Orlando, Tampa • GEORGIA, Atlanta • HAWAII, Honolulu • ILLINOIS, Chicago, Peoria, Rolling Meadows • INDIANA, Indianapolis • IOWA, Bettendorf • KENTUCKY, Louisville • LOUISIANA, New Orleans • MARYLAND, Baltimore, Odenton • MASSACHUSETTS, Boston, Springfield, Waltham • MICHIGAN, Detroit • MINNESOTA, Minneapolis • MISSOURI, Kansas City, St. Louis • NEBRASKA, Omaha • NEW HAMPSHIRE, Manchester • NEW JERSEY, Cherry Hill, Fairfield, Princeton, Somerset • NEW MEXICO, Albuquerque, Los Alamos • NEW YORK, Albany, Buffalo, Long Island, Manhattan, Rochester, Syracuse, Westchester • NORTH CAROLINA, Chapel Hill, Charlotte • OHIO, Cincinnati, Cleveland, Columbus, Dayton • OKLAHOMA, Tulsa • OREGON, Portland • PENNSYLVANIA, Allentown, Harrisburg, Philadelphia, Pittsburgh • RHODE ISLAND, Providence • SOUTH CAROLINA, Columbia • TENNESSEE, Knoxville, Nashville • TEXAS, Austin, Dallas, El Paso, Houston • UTAH, Salt Lake City • VERMONT, Burlington • VIRGINIA, Richmond • WASHINGTON, Seattle • WASHINGTON, D.C. • WEST VIRGINIA, Charleston • WISCONSIN, Milwaukee • INTERNATIONAL —ARGENTINA, Buenos Aires • AUSTRALIA, Adelaide, Brisbane, Canberra, Darwin, Hobart, Melbourne, Perth, Sydney, Tasmania, Townsville • AUSTRIA, Vienna • BELGIUM, Brussels • BOLIVIA, La Paz • BRAZIL, Rio de Janeiro, Sao Paulo • CANADA, Calgary, Edmonton, Halifax, Kanata, London, Montreal, Ottawa, Quebec City, Toronto, Vancouver, Winnipeg • CHILE, Santiago • DENMARK, Copenhagen • EGYPT, Cairo • FINLAND, Helsinki • FRANCE, Lyon, Marseille, Paris, Puteaux • HONG KONG • INDIA, Bombay • IRAN, Tehran • IRELAND, Dublin • ISRAEL, Tel Aviv • ITALY, Milan, Rome, Turin • JAPAN, Osaka, Tokyo • MEXICO, Mexico City • NETHERLANDS, Amsterdam, Hague, Utrecht • NEW ZEALAND, Auckland, Christchurch, Wellington • NORTHERN IRELAND, Belfast • NORWAY, Oslo • PERU, Lima • PUERTO RICO, San Juan • SAUDI ARABIA, Jeddah • SCOTLAND, Edinburgh • SINGAPORE • SOUTH KOREA, Seoul • SPAIN, Madrid • SWEDEN, Gothenburg, Stockholm • SWITZERLAND, Geneva, Zurich • TAIWAN, Taoyuan • UNITED KINGDOM, Birmingham, Bristol, Ealing, Epsom, Leeds, Leicester, London, Manchester, Reading, Welwyn • VENEZUELA, Caracas • WEST GERMANY, Berlin, Cologne, Frankfurt, Hamburg, Hannover, Munich, Nurnberg, Stuttgart • YUGOSLAVIA, Ljubljana •