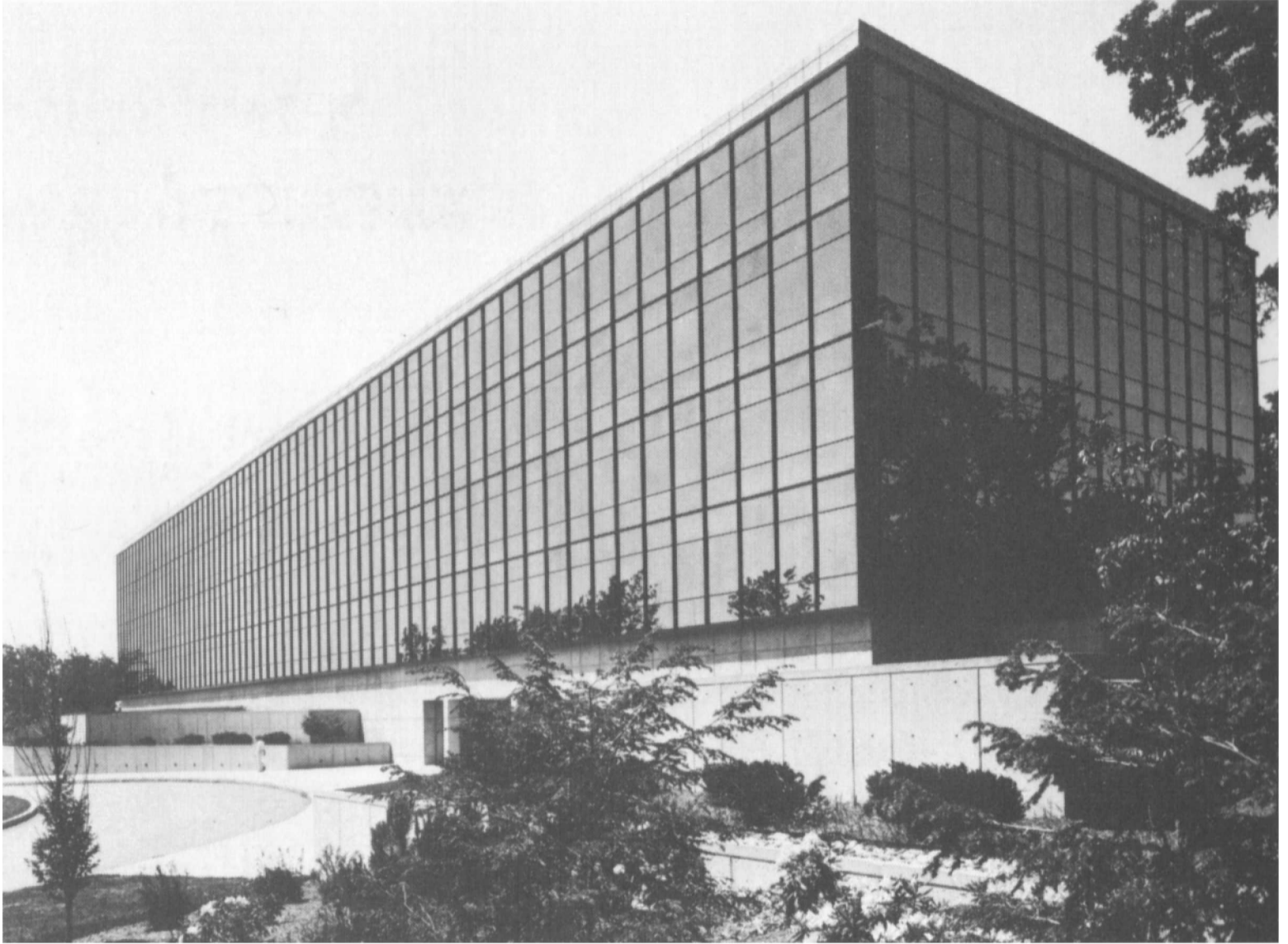


digital

# microcomputers and memories



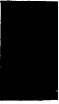


*DIGITAL facility, Marlboro, Massachusetts*

## **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.

**section I  
commonalities**



**section II  
board-level  
components**



**section III  
systems**



**section IV  
appendices**





# **microcomputers and memories**

**digital**

Copyright© 1982 Digital Equipment Corporation.  
All Rights Reserved.

Digital Equipment Corporation makes no representation that the interconnection of its products in the manner described herein will not infringe on existing or future patent rights, nor do the descriptions contained herein imply the granting of license to make, use, or sell equipment constructed in accordance with this description. The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

DEC, DECnet, DECsystem-10, DECSYSTEM-20, DECTape  
DECUS, DECwriter, DIBOL, Digital logo, IAS, MASSBUS, OMNIBUS  
PDP, PDT, RSTS, RSX, SBI, UNIBUS, VAX, VMS, VT  
are trademarks of  
Digital Equipment Corporation

This handbook was designed, produced, and typeset  
by DIGITAL's New Products Marketing Group  
using an in-house text-processing system.

## CONTENTS

### SECTION I - COMMONALITIES

<b>CHAPTER 1</b>	<b>INTRODUCTION</b>	1
LSI-11		1
LSI-11/2		2
LSI-11/23		2
FALCON SBC-11/21		2
PDP-11/23-PLUS		2
THE FAMILY CONCEPT		3
CENTRAL PROCESSOR FEATURES		3
PERIPHERALS		5
SOFTWARE DEVELOPMENT TOOLS		5
RT-11		5
RSX-11M		5
RSX-11M-PLUS		6
RSTS/E		6
RSX-11S		7
RT <sup>2</sup>		8
SIMRT		8
APPLICATIONS		8
DOCUMENTATION		10
EDUCATIONAL SERVICES		11
DECUS		13
MAINTENANCE		13
<b>CHAPTER 2</b>	<b>ARCHITECTURAL OVERVIEW</b>	29
SYSTEM ARCHITECTURE		29
PROCESSOR STATUS WORD		33
INSTRUCTION SET		35
FALCON SBC-11/21 ARCHITECTURAL AND OPERATIONAL FEATURES		36
LSI-11/23 ARCHITECTURAL AND OPERATIONAL FEATURES		36
LSI-11/2 ARCHITECTURAL AND OPERATIONAL FEATURES		37
LEVELS OF INTEGRATION		38

<b>CHAPTER 3 ADDRESSING MODES</b> .....	43
REGISTER MODE .....	46
REGISTER DEFERRED MODE .....	47
AUTOINCREMENT MODE .....	48
AUTOINCREMENT DEFERRED MODE .....	49
AUTODECREMENT MODE .....	49
AUTODECREMENT DEFERRED MODE .....	50
INDEX MODE .....	51
INDEX DEFERRED MODE .....	52
USE OF THE PC AS A GENERAL REGISTER .....	52
PC IMMEDIATE MODE .....	53
PC ABSOLUTE MODE .....	54
PC RELATIVE MODE .....	54
PC RELATIVE DEFERRED MODE .....	55
GRAPHIC SUMMARY OF PDP-11 ADDRESSING MODES .....	59
<b>CHAPTER 4 INSTRUCTION SET</b> .....	65
SINGLE-OPERAND INSTRUCTIONS .....	66
DOUBLE-OPERAND INSTRUCTIONS .....	67
PROGRAM CONTROL INSTRUCTIONS .....	68
JUMP AND SUBROUTINE INSTRUCTIONS .....	69
RTS FORMAT .....	70
TRAPS AND INTERRUPTS .....	70
MISCELLANEOUS INSTRUCTIONS .....	70
CONDITION-CODE OPERATIONS .....	71
EXAMPLES .....	73
SPECIAL SYMBOLS .....	75
INSTRUCTION SET .....	77
EXTENDED INSTRUCTION SET .....	110
<b>CHAPTER 5 FLOATING POINT INSTRUCTION SET (FP-11) (LSI-11/23, PDP-11/23, AND PDP-11/23-PLUS)</b> .....	115
INTRODUCTION .....	115
FLOATING POINT DATA FORMATS .....	116
FLOATING POINT STATUS REGISTER (FPS) .....	118
FLOATING EXCEPTION CODE AND ADDRESS REGISTERS .....	122
FLOATING POINT OPTION INSTRUCTION ADDRESSING .....	123
ACCURACY .....	123
FLOATING POINT INSTRUCTIONS .....	125



<b>CHAPTER 6</b>	<b>FLOATING POINT INSTRUCTION SET (FIS)</b>	
	<b>(LSI-11, LSI-11/2, and PDP-11/03)</b>	153
INTRODUCTION		153
KEV11 OPTION		153
<b>CHAPTER 7</b>	<b>OCTAL DEBUGGING TECHNIQUE</b>	
	<b>(MICROCODE ODT)</b>	161
CONSOLE ODT		161
CONSOLE ODT COMMANDS		161
CONSOLE ODT OPERATION		165
HARDWARE REQUIREMENTS		179
<b>CHAPTER 8</b>	<b>PROGRAMMING TECHNIQUES</b>	185
POSITION-INDEPENDENT CODE		185
EXAMPLES		187
STACKS		189
DELETING ITEMS FROM A STACK		195
SUBROUTINE LINKAGE		195
INTERRUPTS		196
RE-ENTRANCY		200
COROUTINES		203
RECURSION		208
PROCESSOR TRAPS		210
TRAP INSTRUCTIONS		211
CONVERSION ROUTINES		213
ASCII CONVERSIONS		217
<b>CHAPTER 9</b>	<b>LSI-11 BUS</b>	219
DATA TRANSFER CYCLES		223
DIRECT MEMORY ACCESS		234
INTERRUPTS		236
CONTROL FUNCTIONS		242
LSI-11 BUS ELECTRICAL CHARACTERISTICS		244
SYSTEM CONFIGURATIONS		248
MODULE CONTACT FINGER IDENTIFICATION		250

<b>CHAPTER 10</b>	<b>MEMORY MANAGEMENT</b>	265
MEMORY RELOCATION		267
PROTECTION		271
PAGE ADDRESS REGISTER (PAR)		276
PAGE DESCRIPTION REGISTER (PDR)		276
VIRTUAL AND PHYSICAL ADDRESSES		279
STATUS REGISTERS		282
MEMORY MANAGEMENT INSTRUCTIONS		284

<b>CHAPTER 11</b>	<b>FPF11 FLOATING POINT PROCESSOR</b>	287
FEATURES-BENEFITS		287
SPECIFICATIONS		287
CONFIGURATION		288
DESCRIPTION		291

## SECTION II-BOARD LEVEL COMPONENTS

<b>CHAPTER 12</b>	<b>FALCON SBC-11/21 MICROCOMPUTER</b>	299
INTRODUCTION		299
FEATURES-BENEFITS		299
SPECIFICATIONS		300
CONFIGURATION		302
DESCRIPTION		314

<b>CHAPTER 13</b>	<b>LSI-11/23 MICROCOMPUTER</b>	321
INTRODUCTION		321
FEATURES-BENEFITS		321
SPECIFICATIONS		323
ADDITIONAL SOURCES OF LSI-11/23 DOCUMENTATION		323
CONFIGURATION DATA-JUMPER SELECTION		323
DESCRIPTION		331
LSI-11/23 PROCESSOR OPTIONS (KEF11-AA)		350
FPF11		351

<b>CHAPTER 14</b>	<b>LSI-11/2 MICROCOMPUTER</b> .....	353
INTRODUCTION	.....	353
FEATURES-BENEFITS	.....	353
SPECIFICATIONS	.....	354
ADDITIONAL SOURCES OF LSI-11/2 DOCUMENTATION	.....	355
CONFIGURATION DATA	.....	355
JUMPER SELECTION	.....	356
DESCRIPTION	.....	360
LSI-11/2 PROCESSOR OPTIONS (KEV11)	.....	374
<b>CHAPTER 15</b>	<b>LSI-11 MICROCOMPUTER</b> .....	377
INTRODUCTION	.....	377
FEATURES-BENEFITS	.....	377
SPECIFICATIONS	.....	378
ADDITIONAL SOURCES OF LSI-11 DOCUMENTATION	.....	379
CONFIGURATION DATA-JUMPER SELECTION	.....	379
LSI-11 PROCESSOR OPTIONS (KEV11)	.....	386
<b>CHAPTER 16</b>	<b>LSI-11 SYSTEM TROUBLESHOOTING</b> .....	389
INTRODUCTION	.....	389
CONDITION ONE	.....	391
CONDITION TWO	.....	393
NOP PROGRAM	.....	398
LTC PROGRAM	.....	398
INTERRUPT TEST	.....	400
RELATED MICROCOMPUTER AIDES	.....	401
<b>CHAPTER 17</b>	<b>ROM MEMORIES</b> .....	405
MRV11-AA	.....	405
MRV11-BA	.....	416
MRV11-C	.....	432

<b>CHAPTER 18 PROMS</b> .....	461
PROGRAMMING NOTES .....	461
USING PROMS .....	462
PB11 UNIVERSAL PROM PROGRAMMER .....	470

<b>CHAPTER 19 RAM MEMORIES</b> .....	475
MSV11-B .....	475
MSV11-CD .....	479
MSV11-D,E .....	485
MSV11-L .....	506
MCV11-D .....	530

<b>CHAPTER 20 MULTIFUNCTION MODULES</b>	
<b>MXV11-A MEMORY AND SYNCHRONOUS</b>	
<b>SERIAL LINE INTERFACE</b> .....	541
INTRODUCTION .....	541
FEATURES-BENEFITS .....	541
SPECIFICATIONS .....	542
CONFIGURATION .....	543
MXV11 TU58 BOOTSTRAP .....	563
MXV11 DISK BOOTSTRAP .....	564
BAUD RATE JUMPERS .....	565
CABLE DEFINITIONS .....	567
DESCRIPTION .....	569

**SECTION III-SYSTEMS**

<b>CHAPTER 21 SYSTEMS</b> .....	583
FUNCTIONING AND PERFORMANCE .....	584
PDP-11/23 PLUS PACKAGED SYSTEMS .....	589
PDP-11/23 AND PDP-11/03-L PACKAGED SYSTEMS .....	595
SYSTEM EXPANSION .....	601
PROCESSOR SELF-DIAGNOSIS .....	603

<b>CHAPTER 22</b>	<b>PDP-11/23-PLUS MICROCOMPUTER</b>	605
INTRODUCTION		605
FEATURES-BENEFITS		605
SPECIFICATIONS		606
DESCRIPTION		607
CONFIGURATION		622
<b>CHAPTER 23</b>	<b>MSV11-P MEMORY</b>	641
FEATURES-BENEFITS		641
SPECIFICATIONS		642
CONFIGURATION		644
DESCRIPTION		649
<b>APPENDIX A</b>	<b>ASSIGNMENT OF ADDRESSES AND VECTORS</b>	663
<b>APPENDIX B</b>	<b>LSI-11/23 INSTRUCTION TIMING</b>	681
<b>APPENDIX C</b>	<b>LSI-11, /2 INSTRUCTION TIMING</b>	695
<b>APPENDIX D</b>	<b>PDP-11 FAMILY DIFFERENCES</b>	701
<b>APPENDIX E</b>	<b>INTEGRATED CIRCUITS</b>	723
<b>APPENDIX F</b>	<b>FALCON SBC-11/21 INSTRUCTION TIMING</b>	727
<b>APPENDIX G</b>	<b>LSI-11 DOCUMENTATION</b>	733
<b>APPENDIX H</b>	<b>PDP-11/23-PLUS INSTRUCTION TIMING</b>	741
<b>INDEX</b>		749



## PREFACE

Today is the advent of the Microcomputer age.

DIGITAL maintains and continues to expand its strong leadership in the 16-bit, board-level microcomputer marketplace. With its introduction in November 1981 of the FALCON SBC-11/21, and succeeding announcement of the systems-oriented PDP-11/23- PLUS, DIGITAL is determined to provide the computer user, whose system design is stretching the capabilities of 8-bit performance, advanced 16-bit capability.

This Handbook, having undergone extensive revision from its predecessor, serves a dual role in addressing both the needs of the board-level customer and the systems-oriented user. It offers solutions to a very wide range of user application requirements which can be met by both DIGITAL stand-alone microcomputer systems, as well as board-level component microcomputer products. We have designed this Handbook to assist those customers desiring a descriptive, detailed tutorial on the benefits, specifications, design, configuration and functionality of DIGITAL's maturing family of board-level and systems-level LSI-11 microcomputers. Although primarily aimed at experienced computer users, first-time microcomputer users will find some of the Handbook tutorial information helpful.

The 1982 Micrometers and Memories Handbook is divided into three major sections. The first section, called the 'Commonalities' section, encompasses information pertaining to both systems and boards. The second section, or Section II, deals strictly with board-level component subject matter, while the third section, Section III, presents material related solely to stand-alone systems.

The Commonalities section is uniquely constructed to provide a step-by-step guide to lead the user through the introductory passages such as the Introduction, Architectural Overview, and Addressing Modes to the more complex Instruction Set chapters, which have been divided conveniently into three separate chapters. These include: The basic PDP-11 instruction set chapter, which is common to all DIGITAL microcomputers; the Floating Point (Option) Instruction Set (FP-11), available for use with the LSI-11/23, PDP-11/23, and the PDP-11/23-PLUS; and the Floating Point Instruction Set (FIS) to be used with the LSI-11, LSI-11/2, and the PDP-11/03.

Highly technical, detailed configuration information on specific microcomputers and memories follows in Section II. The individual LSI-11 microcomputer chapters have been reconstructed, with the configuration information positioned towards the beginning of each chapter, to provide immediately accessible reference information on the configuration of the microcomputer. An upgraded features and benefits section illustrates the most current information on the individual capabilities of these microcomputers. Likewise, the chapters on memories have been divided into two separate sections: Random Access Memories (RAM) and Read-Only Memories (ROM). DIGITAL's newest memory, the MSV11-L, is a single-board memory with byte parity and 64 KB MOS RAM memory technology. The MCV11-D, DIGITAL'S new CMOS read/write battery backup memory offering, is also featured. Another new system-related memory, the MSV11-P, is featured in the Memories chapter in Section III. These memory chapters have been redesigned to reflect the same effective outline as the microcomputer chapters.

Section III also presents for the first time in a DIGITAL Microcomputer Handbook a specific chapter on a systems-based PDP-11 microcomputer--the PDP-11/23-PLUS. A significantly expanded chapter on DIGITAL's systems offerings is also featured in this section.

Among this Handbook's more intermediary-level chapters, chapters that require an advanced knowledge of microcomputer concepts due to their highly sophisticated, technically detailed contents, are the Memory Management, Programming Techniques, Octal Debugging Techniques, and the LSI-11 Bus chapters. These chapters appear in Section I. The Memory Management and LSI-11 Bus chapters highlight one of DIGITAL's newest technological achievements-- Q-bus 22-bit addressing. This feature provides the LSI-11/23 with the means to address up to four megabytes of memory. The Octal Debugging Technique (ODT) chapter introduces the ODT as a significant tool which facilitates the running and debugging of programs.

Additional material in this Handbook include helpful new chapters covering system procedural troubleshooting for general LSI-11-based microcomputer systems and the floating point processor, which speeds execution of the floating-point arithmetic instructions.

Lastly, the Appendices in the back of this Handbook contain the most current, accurate and complete support data and timing available at the time of publication.

In late 1982 DIGITAL will publish a document that will assist technically-oriented readers familiar with fundamental computer concepts. Written in easily understood, clear, and very simple narrative, this



Handbook describes in detail the process of designing a microcomputer-based product, using DIGITAL board-level microcomputers as primary examples. Also, general information will be provided to teach readers how to use our microcomputers and describe the benefits derived from their usage. In addition, this Handbook will cover such topics as designing application software, specifying application requirements, interfacing DIGITAL hardware, and programming in the development system.



**section I**  
**commonalities**





## CHAPTER 1 INTRODUCTION

DIGITAL microcomputers combine the power, compatibility, and experience of the PDP-11 minicomputer family with advanced LSI technology and compact, modular design. Together, this combination establishes a standard for efficiency and ease-of-use, backed by more than six years of extensive microcomputer development and experience.

The DIGITAL 16-bit microcomputer family includes LSI-11s on boards: LSI-11, LSI-11/2, LSI-11/23, and its newest and first single-board microcomputer—the FALCON SBC-11/21. PDP-11s utilize the LSI-11 CPU boards, but are offered either as rack-mountable boxes or complete packaged systems. These are the PDP-11/03, PDP-11/23, and the PDP-11/23-PLUS. Both board-level and systems-level microcomputer family members, with the exception of the PDP-11/03 and the PDP-11/23, are discussed individually in the user-level, functionally detailed chapters that follow in this Handbook. PDP-11/03 processor capabilities are the same as the LSI-11 or the LSI-11/2, and the PDP-11/23 has the same capabilities as the LSI-11/23.

Except when otherwise specifically noted throughout the Commonalities section of this Handbook, LSI-11 microcomputers and PDP-11 microcomputers should be considered interchangeable—that is, for example, whenever LSI-11/23 appears, a PDP-11/23 microcomputer offers the same capabilities. In both cases, the PDP-11s and LSI-11s discussed in this Handbook have a very essential feature in common—the LSI-11 BUS.

### LSI-11

In 1975, the DIGITAL LSI-11 microcomputer, called the "computer-on-a-board", became the first microcomputer in history to utilize 16-bit architecture. This architecture characteristic only of minicomputers at

---

the time, set a standard for the microcomputer industry. The LSI-11 microcomputer's quad-height size of 10" × 8.5" (26 cm × 22 cm) provided easy installation into a variety of environments and applications. It featured 8 KB of resident RAM and complete functional compatibility with DIGITAL's PDP-11 family of minicomputers.

### **LSI-11/2**

DIGITAL announced the LSI-11/2 microcomputer in 1977. The LSI-11/2 retained the same processor features as its predecessor—the LSI-11. However, the board dimensions were reduced from 10" × 8.5" (26 cm × 22 cm) to the double-height size of 5.2" × 8.5" (13 cm × 22 cm). The result was a microcomputer board that was even easier to handle, stock, and package into a wider range of hardware environments. The address range of the LSI-11/2 is 64 KB.

### **LSI-11/23**

The LSI-11/23 microcomputer, introduced in 1979, utilized state-of-the-art Large Scale Integration (LSI) technology. Developments in N-MOS technology made this possible. This LSI-11/23, while retaining the LSI-11/2 microcomputer form factor, increased CPU performance to 2.5 times faster than speeds of the LSI-11 and the LSI-11/2, thereby approaching execution speeds of mid-range minicomputers.

In 1981, further functionality was added to the LSI-11/23 microcomputer. An extended 22-bit addressing range capability was introduced, making it possible for the LSI-11/23 microcomputer to address four megabytes of main memory. The comprehensive memory management feature provides memory relocation, segmentation, and protection for this extended address range. Together, these features are not found in any other microcomputer today. The LSI-11/23 microcomputer supports RSX-11M, the operating system that has set a worldwide standard for real-time minicomputer performance.

### **FALCON SBC-11/21**

November 1981 marked a very significant milestone for DIGITAL. The FALCON SBC-11/21, DIGITAL's first single-board, lowest-cost microcomputer was announced. This 16-bit, low-end microcomputer, designed exclusively for dedicated, ROM-based, real-time applications offers more on-board RAM and ROM memory capability than any DIGITAL microcomputer. It also features two asynchronous serial I/O ports with eight programmable baud rates, 24 parallel I/O lines, and a crystal-controlled real-time clock. In addition, the FALCON offers a very compact packaging area. Its double-height, 44 square inch board size is the smallest in the microcomputer industry.

### **PDP-11/23-PLUS**

Shortly after the announcement of the FALCON in November 1981, DIGITAL introduced another new microcomputer—the PDP-11/23-PLUS. The PDP-11/23-PLUS, while providing full PDP-11 functionality, is offered in systems supporting up to one megabyte of parity MOS

memory. The PDP-11/23-PLUS's compact packaging density offers more functionality using the same volume space in the processor box as the PDP-11/23.

### **THE FAMILY CONCEPT**

The LSI-11/2, LSI-11/23, and FALCON SBC-11/21 offer a wide range of solutions to meet the ever-changing needs of the board-level user. Similarly, the PDP-11/03, PDP-11/23, and PDP-11/23-PLUS provide complete box-and systems-level solutions to meet the requirements of end-users and OEMs. Collectively, LSI-11 and PDP-11 microcomputer 16-bit architecture allows easier programming, greater I/O throughput, and more flexibility than the older 8-bit microcomputers.

In keeping with the DIGITAL philosophy of PDP-11 compatibility, DIGITAL's microcomputers have many common features that represent years of experience in computer technology.

These design features were kept consistent to protect your hardware and software investments by making growth and migration easier for your product.

### **CENTRAL PROCESSOR FEATURES**

CPU features common to all DIGITAL LSI-11 and PDP-11 microcomputers include:

- *The PDP-11 Instruction Set*—87 standard PDP-11 instructions comprise the basic instruction set of the LSI-11 microcomputer family. These instructions process bits, bytes, and words, as well as sub-routines, interrupts, single-operands, and double-operands.
- *Six General-Purpose Registers, Two Special Purpose Registers*—For further versatility, the LSI-11s contain six general-purpose registers for use as accumulators, address pointers, index registers, and other specialized functions. The two special purpose registers are utilized for the Stack Pointer (SP) and the Program Counter (PC).
- *Twelve Addressing Modes*—The addressing modes, common to all LSI-11s and PDP-11s, allow the extension of the basic PDP-11 Instruction Set to create over 400 powerful instruction combinations, as well as byte and word addressing. This allows user programs to be structured around bytes, words, or mixed, according to the user's needs.

The programmer can access and manipulate data from memory directly or indirectly. The addressing modes consist of register addressing, autoincrement or autodecrement, and indexed addressing. All addressing modes can be direct or deferred.

In addition, four of these modes can be used with the program counter to provide immediate, absolute, relative, and relative deferred addressing. The addressing modes provide flexible and efficient programming.

- *The Extended Instruction Set (EIS)*—Optional on the LSI-11 and LSI-11/2, standard on the LSI-11/23, the EIS contains four instructions that perform signed integer multiply and divide and direct implementation of multiple shifting. (The EIS is not available on the FALCON SBC-11/21 microcomputer).
- *The LSI-11 Bus*—Common to all LSI-11 processors and peripherals, this bus provides vectored priority interrupts, programmed I/O transfers, and DMA I/O data transfers. All modules operate asynchronously at their highest possible speed.
- *Power-fail/Auto Restart Logic*—This feature initiates an automatic orderly system shutdown by generating a microcode power-fail sequence when power sequencing signals from the power sequence module or system power supply indicate a possible AC power loss.

When power is restored, a complementary sequence in microcode permits the processor to begin operating in one of four modes. This procedure allows the program to continue at either the point of interruption or, in some cases, reboots the system.

- *ODT/ASCII Console Routine*—Optional on the FALCON microcomputer, standard on the LSI-11, LSI-11/2, and LSI-11/23, the octal debugging function allows the conventional front panel to be replaced by any terminal device generating ASCII code. The Octal Debugging Technique feature can also be used to implement remote control and/or loading of an LSI-11 without requiring the use of a separate ROM program.
- *A Real-Time Clock Input*—This feature provides synchronization of the CPU with real-time events such as power-line frequencies.
- *Optional Microcoded Floating Point*—Floating point operations can be performed on LSI-11 microcomputers via a Floating Point Instruction Set (KEV11) for the LSI-11 and LSI-11/2 and a more comprehensive Floating Point Option (KEF11) for the LSI-11/23. (A floating point option is not available for the FALCON microcomputer).
- *Optional Floating Point Processor*—A floating point processor (FPF11) is also available for the LSI-11/23 to increase the execution speeds of floating point instructions. These instructions are the same instructions implemented in the floating point option (KEF11).



## **PERIPHERALS**

DIGITAL manufactures a wide range of peripheral equipment to meet specific customer needs in business, education, industry, laboratory, and medicine. I/O storage devices range from tape cartridges to high-volume disk packs, and from the DECwriter to intelligent terminals that provide both hard copy and video display.

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

## **SOFTWARE DEVELOPMENT TOOLS**

### **RT-11**

The LSI-11/2 and LSI-11/23 systems both run the RT-11 floppy- or hard-disk-based, real-time, foreground/background or single-job operating system. The RT-11 operating system is designed for the single interactive user, and offers a wide range of industry standard high-level and assembly languages for program development.

RT-11's easy-to-use command language and straightforward status messages provide a "friendly" user interface. The compactness and efficiency of the RT-11 real-time software system allow it to run on the lowest-cost hardware configurations. RT-11 can support a single-job monitor where one program resides in memory at one time and runs until it is completed or interrupted with a keyboard command. Or, RT-11 can support real-time job execution in the foreground, while simultaneously performing an interactive or batch program development job in the background.

In 1981, DIGITAL introduced MicroPower/Pascal, a new software product developed to address the needs of programmers seeking to write technical microcomputer applications. MicroPower/Pascal is comprised of a new language processor, a modular operating system for the target application, and powerful debugging tools for use in the program development environment. Program development for MicroPower/Pascal is implemented under RT-11.

### **RSX-11M**

RSX-11M, the hard-disk-based, multiuser, multitasking, real-time operating system, is available for use on the LSI-11/23. The LSI-11/23's greater memory capacity and memory management hardware complement features present in RSX-11M. The benefits of these extra hardware features provide larger, more sophisticated development programs for system users.

RSX-11M also allows programs to be partitioned and segmented in main memory for fast and efficient access by the high-performance

LSI-11/23 CPU, and provides complete protection from any unauthorized user. Additionally, code and data sharing reduce memory requirements, allowing for more programs in memory.

RSX-11M also offers task checkpointing; a program or task currently running memory can be interrupted and swapped out of memory to a disk when a higher-priority task requests the partition in which it is resident. After the priority task completes execution, the checkpointed task is automatically returned to memory and re-stored at the point where it was interrupted.

### **RSX-11M-PLUS**

The RSX-11M-PLUS, a superset of the RSX-11M operating system, expands the capability of the RSX-11M to provide ease-of-use while optimizing large memory systems. RSX-11M-PLUS is now available for use with the PDP-11/23-PLUS.

### **RSTS/E**

RSTS/E, Resource Sharing Timesharing System/Extended, is a highly interactive, multiuser, multitasking, general-purpose operating system. RSTS/E dynamically allocates system resources such as processor time, memory space, file space, and peripherals on a best fit/best throughput basis to continually keep processing efficient. RSTS/E is now available on the PDP-11/23 and the PDP-11/23-PLUS microcomputer systems.

### **Layered Products**

DIGITAL offers a wide variety of layered products such as DECnet, DECnet/E, and DECword.

### **Languages**

Fully supported languages available for use with RSX-11M and RT-11 include:

- *MACRO-11*—A symbolic assembly language that features powerful macro capabilities. Program readability and maintainability are enhanced through symbolic names representing relocatable addresses, constants, or a series of machine instructions.
- *FORTRAN IV*—A superset of the ANSI standard, FORTRAN IV has extra features that allow easier debugging of existing FORTRAN programs. Further extensions to ANSI standard FORTRAN IV allow shorter development time for new software.

The FORTRAN IV compiler produces binary code segmented into read-only and read/write sections, and is suitable for placing in ROM.

- **BASIC-11**—An extension of the popular Dartmouth-developed BASIC language, its interpretive nature allows for quick software development and debugging. Sophisticated programs can take advantage of a set of language extensions that permit advanced manipulation of mass storage and terminal devices.

#### **Additional Languages Supported Under RT-11**

- **APL-11**—A mathematically structured programming language that features multiple statement lines, standard file-naming formats and the ability to fully evaluate character strings and write user-defined functions.
- **MicroPower/Pascal**—Applications utilizing MicroPower/Pascal are developed under RT-11 in Pascal, a superset of the Jensen and Wirth definition of Pascal. The Pascal compiler is highly efficient and produces optimized code as well as ROM/RAM separation of code.
- **Multi-user BASIC**—A multi-user environment which supports up to eight users performing interactive software development and debugging in BASIC-11.

#### **Additional Languages Supported Under RSX-11M**

- **FORTRAN IV-PLUS**—A FORTRAN IV superset that features highly optimized compiler-generated code that fully utilizes the power of the LSI-11/23's optional Floating Point Unit. This hardware/software combination boosts performance in floating point intensive applications.
- **BASIC-PLUS-2**—A superset of the BASIC-11 language that contains compiled code, call statements, record I/O, and interactive debugging.

#### **Run-Time Application Support**

MicroPower/Pascal, RSX-11S, RT<sup>2</sup>, and SIMRT provide various levels of run-time applications support as summarized below.

**MicroPower/Pascal** — The MicroPower/Pascal operating system which runs in the target application is modular, ROM(able), and supports the entire range of microcomputer hardware products, including the LSI-11/23 and FALCON SBC-11/21. Due to its modularity, the minimum MicroPower/Pascal operating system is significantly smaller than traditional operating systems.

**RSX-11S** — A memory-resident, execute-only subset of the RSX-11M operating system that can handle up to 256 independent tasks through multiprogramming priority scheduling. Because it is memory based, RSX-11S is not dependent on any mass storage device for

execution. System generation and program development must take place in a RSX-11M development system.

System generation produces a system/task image that can be booted on the target system, or down-line loaded via DECnet, DIGITAL's networking software. Further additional tasks can be transported to a running RSX-11S system via storage media or DECnet.

**RT<sup>2</sup>** — is a license to copy a subset of RT-11 that supports an execution-only mass storage based target system. In addition to foreground/background processing, RT<sub>2</sub> includes a license to use standard RT-11 utility programs for file creation, deletion, transfer, and renaming. Program development occurs on an RT-11 host system and produces a self-starting application system based on a mass storage medium.

**SIMRT** — is available as part of the RT-11 FORTRAN IV package and provides stand-alone support for FORTRAN programs. SIMRT can be linked with a FORTRAN program that has been developed on an RT-11 host system to provide a bootable image. This image can be transferred to the target system in one of three ways:

1. The image can be placed in ROMs on the target system.
2. The image can be down-line loaded into the target system.
3. The image can be booted from a mass storage device on the target system.

## **APPLICATIONS**

Over 100,000 LSI-11 family microcomputers and their associated products are in worldwide use today in a variety of applications. Many users are reaping the benefits of low initial investment costs that generate early revenue flows, because their packaged products are ready for market sooner.

The LSI-11 products can evolve and expand without major design changes. The easy system integration and maintenance afforded by the LSI-11 family provide the necessary tools that easily accept extra custom-designed enhancements and added value for specific application areas. Industrial process control, OEM hardware, and turnkey commercial and end-user systems are just a few of many general application areas ideal for LSI-11 microcomputers.

### **Industrial Process Control**

In this application area, users have packaged LSI-11s to meet a variety of process control requirements. Manually operated process control

**Table 1-1 Summary/Comparison Chart of Specifications**

FEATURE	LSI-11/2	LSI-11/23	SBC-11/21	PDP-11/23-PLUS
FORM FACTOR	5.2" x 8.9" 13.2 cm x 22.8 cm			10.5" x 8.9" 26.6 cm x 22.8 cm
INTERRUPTS	1 LEVEL (LEVEL 4)	4 LEVELS (LEVELS 4, 5, 6, 7)	4 LEVELS (LEVELS 4, 5, 6, 7)	4 LEVELS (LEVELS 4, 5, 6, 7)
LSI-11 BUS STRUCTURE	16-BIT ADDRESS 16-BIT DATA 1 LEVEL INTERRUPT DMA ASYNCHRONOUS PARALLEL	22-BIT ADDRESS 16-BIT DATA 4 LEVEL INTERRUPT DMA ASYNCHRONOUS PARALLEL	16-BIT ADDRESS 16-BIT DATA 4 LEVEL INTERRUPT DMA ASYNCHRONOUS PARALLEL	22-BIT ADDRESS 16-BIT DATA 4 LEVEL INTERRUPT DMA ASYNCHRONOUS PARALLEL
ADDRESSING RANGE	64 KB	4 MB**	64 KB	1 MB
BASIC INSTRUCTIONS	>400			
MEMORY MANAGEMENT	NOT APPLICABLE	FULL FEATURE. RELOCATION SEGMENTATION. PROTECTION (OPTIONAL)	NOT APPLICABLE	FULL FEATURE. RELOCATION SEGMENTATION. PROTECTION (STANDARD)
MEMORY TYPES	MOS PROM/ROM/RAM			
FLOATING POINT	OPTIONAL (FIS) KEV11	OPTIONAL FPF11 or KEF11	NOT APPLICABLE	OPTIONAL FPF11 or KEF11
EXTENDED INSTRUCTION SET (EIS) (MUL, DIV, ASH, ASHC)	OPTIONAL	STANDARD	NOT APPLICABLE	STANDARD
OPERATING MODES	NOT APPLICABLE	KERNEL AND USER WITH (MMU)	NOT APPLICABLE	KERNEL AND USER
POWER CONSUMPTION	+5V 1.2A +12V 0.22A	+5V 2.0A +12V 0.2A	+5V 2.8A +12V 1.10A	+5V 4.5A +12V 0.3A
BUS LOADS	AC 1.7 UNIT LOADS DC 1 UNIT LOAD	AC 2 UNIT LOADS DC 1 UNIT LOAD	AC 2 UNIT LOADS DC 1 UNIT LOAD	AC 2 UNIT LOADS DC 1 UNIT LOAD
ENVIRONMENTAL TEMPERATURE	OPERATING: 5° to 60° C (41° to 140° F) STORAGE: -40° to 66° C (-40° to 151° F)	OPERATING: 5° to 60° C (41° to 140° F) STORAGE: -40° to 65° C (-40° to 149° F)	OPERATING: 5° to 60° C (41° to 140° F) STORAGE: -40° to 65° C (-40° to 149° F)	OPERATING: 5° to 60° C (41° to 140° F) STORAGE: -40° to 65° C (-40° to 149° F)
MICRO CYCLE TIME	380 ns	290 ns	610 ns	300 ns
RELATIVE SPEED	1.0	2.5	2.0*	2.5
DEVELOPMENT SOFTWARE	RT-11	RT-11, RSX-11M	NOT APPLICABLE	RT-11, RSX-11M RSX-11M-PLUS, RSTS/E
RUN-TIME SERVICES	RSX-11S, RT2, SIMRT, MicroPower/Pascal (only under SBC-11/11/21)			

\*When executing on-board memory.

\*\*Depends upon backplane.

systems have been replaced with cost-effective LSI-11s to improve the quality of materials produced.

Table 1-1 summarizes and compares the features of the LSI-11/2 with those of the LSI-11/23, FALCON SBC-11/21, and the PDP-11/23-PLUS.

LSI-11s, with PDP-11s in some instances, are solving data acquisition and control problems through remote control and by monitoring communications via radio links. LSI-11 users in other process control areas, such as automatic system testing, are experiencing reduced test times, improved repetition of test measurements, and the automatic documentation of test sheet procedures.

### **OEM Hardware/Software**

LSI-11s are found in many application areas through OEM custom-designed hardware and software packages. In X-ray analysis they are used with scanning electron microscopes, transmission scopes or microprobes for rapid non-destructive determination of sample composition. They are automating the collection and analysis of generated X-rays to identify elements present with the use of special computer programs that provide accurate weight percentage readouts.

In voice data-entry systems, LSI-11s are packaged with OEM voice data-entry terminals and DIGITAL floppy disks for wholesale cash-and-carry warehouses. The warehouse checker calls out a customer's account number and the product codes of the goods as they move through the checkout area. He inputs the data via a microphone to a voice recognition system controlled by an LSI-11 microcomputer.

The LSI-11 recognizes the speech patterns through signals from the voice terminal as being valid command words and initiates a DECwriter, stationed at a central cashier's collecting point, to print out an invoice, providing up-to-date inventory status and accurate billing.

### **Data Systems**

Data Systems offer hardware and software configurations to address application needs and are optimized for small businesses, distributed processing, and commercial environments. A large amount of commercial software is available from DIGITAL and OEMs to end-users to provide solutions to their application problems.

### **DOCUMENTATION**

DIGITAL offers several levels of technical documentation describing LSI-11 family software and hardware. *The Microcomputer Handbook Series* presents user-level technical documentation for the LSI-11 family. Hardware technical manuals offer 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 covered in commercially available books. If you have a specific documentation need, discuss the issue with a DIGITAL salesperson, who can recommend the appropriate literature.

## **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 to support all DIGITAL systems.

Catalog courses are regularly scheduled classes offered at training centers 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 can 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. Onsite 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 students' 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 students to progress at their own rates, study when and where they wish, and play back modules for review. Audio-visual course material is available in several forms—videotape, videocassette, or audio/filmstrip cassette—all supported by student workbooks.

**LSI-11 Related Courses**

DIGITAL's Educational Services group offers a series of courses on the hardware and software of your LSI-11 system. For complete information on course content, prerequisites, pricing, and scheduling, consult the DIGITAL Education Courses Catalog.

**Boston area:**

Digital Equipment Corporation  
Educational Services Department  
12 Crosby Drive  
Bedford, Massachusetts 01730  
Telephone: (617)-275-5000

**Canada area:**

Digital Equipment France  
Boulevard de France  
France Evry-Tour Lorraine  
F-91000 Evry  
France  
Telephone: 1/077-9000

**West Germany area:**

Digital Equipment GmbH  
Wallensteinplatz 2  
D-8000 Munich 40  
West Germany  
Telephone: 089/3503-1

**Japan area:**

Digital Equipment Corporation  
International-Japan  
Kowa Building No. 25 (3rd Floor)  
8-7, San bancho, Chiyoda-KU  
Tokyo 102  
Japan  
Telephone: (813) 164-7107

**Netherlands area:**

Digital Equipment B.V.  
Kaap Hoorndreef 38  
NL-3563 AV Utrecht, Holland  
Telephone: 030/63-12-12

**San Francisco area:**

Digital Equipment Corporation  
Educational Services Department  
2525 Augustine Drive  
Santa Clara, California 95051  
Telephone:  
(408) 727-0200 Ext. 2142

**Australia And**

**New Zealand:**

DECUS  
P.O. Box 491  
Crows Nest, N.S.W. 2065  
Australia

**Canada:**

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

**Europe and**

**Middle East:**

DECUS  
Case Postale 340  
1211 Geneva 26  
Switzerland

**All Others:**

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



## **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 sponsors 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:

## **MAINTENANCE**

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

### **Onsite Service**

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

- *Service Agreement* — Onsite 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 materials. Users have a choice of tailored service agreements offered by DIGITAL. In addition to basic coverage, extended hours are available to customers with the most critical applications that require special attention. Please refer to

DIGITAL's International Directory of Services for more information. The order number is EJ01333-94 05.

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

### **DIGITAL Software Services**

DIGITAL's Software Services specialists help customers determine the right computer system for their application. Software support continues, through installation, warranty services, and post-warranty contractual services, to assist customers with every phase of their systems' analysis, design, and implementation. For products that require it, a DIGITAL software specialist installs the software and verifies that the system is complete. The specialist also familiarizes the staff with the system's operation and explains its capabilities and documentation. DIGITAL's professional expertise is available through both resident and per call arrangements. Software specialists provide advice on system analysis, application design review, optimization, and system/application integration. Further manpower resources are available to perform specific project tasks or supplement a customer's programming staff.

### **Offsite 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 offsite repairs for North American customers. The CRA assures the customer of complete "one-stop shopping" for all factory-level warranty and post-warranty services. All repairs are made at our Module Repair Facility in Woburn.

For European, Australian, and Japanese customers, we have established Product Repair Centers (PRCs) in eleven countries. Customers can return defective materials 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 offsite repair services. These centers should be contacted for all offsite 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, Massachusetts 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, Belgium  
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

**West Germany**

Product Repair Center Manager  
Digital Equipment GmbH  
Ingoldstaedterstrasse 62  
Euro Industrie Park 2  
D-8000 Munich 45  
West Germany  
Telephone: (89) 31780

**Netherlands**

Product Repair Center Manager  
Digital Equipment Bv  
Coloradodreef 26-28  
P.O. Box 9064  
NI-3563 AV 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) 730-08-000

**Switzerland**

Product Repair Center Manager  
Digital Equipment Corp. AG/SA  
Schaffhauserstrasse 144  
CH-8352 Kloten/24  
Switzerland  
Telephone: (01) 816-9111

**United Kingdom**

Product Repair Center Manager  
Digital Equipment Corp., Ltd.  
Gasworks Road  
Building V.7.A.B.L. Site  
Reading RG1-3EF  
England  
Telephone: (734) 58 35 55

**General International Area**

At this time, the only services offered in the GIA are 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. Australia 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

Table 1-2 Module Specifications

Option Desig.	Module No(s).	Description	Power Requirements		Bus Loads*		Size
			+ 5V ±5%	+ 12V ±3%	AC(Max)	DC	
AAV11-A	A6001	4-channel, 12-bit D/A converter	1.5 A	0.4 A	1.9	1	Quad
AAV11-C	A6006	4-channel, 12-bit D/A converter	2.0 A	—	0.9	1.0	Double
ADV11-A	A012	16-channel, 12-bit A/D converter	2.0 A	0.45 A	3.25	1	Quad
ADV11-C	A8000	16 single-ended or 8 differential A/D channels, 12-bit	1.5 A	—	1.3	1.0	Double
AXV11-C	A0026	Analog I/O board 16 single-ended analog input channels, 12-bits 2 D/A output, 12-bit channels	1.5 A	—	1.3	1.0	Double

\* These ac load figures were measured using standard TDR (time domain reflectometry) techniques. The conversion factor is 9.35 pF/ac load. These numbers are nominal values which will tend to vary from module to module due to normal tolerances of components used in the manufacturing of the product.

Table 1-2 (cont) Module Specifications

Option Desig.	Module No(s).	Description	Power Requirements		Bus Loads*		Size
			+ 5V ±5%	+ 12V ±3%	AC(Max)	DC	
BDV11	M8012	Bootstrap, terminator, diagnostic	1.6 A	0.07 A	2.0	1	Quad
DDV11-B DLV11	M7940	6 X 9 backplane Asynchronous serial line interface	1.0 A	0.18 A	6.4 2.5	0 1	Double
DLV11-E	M8017	Asynchronous line interface	1.0 A	0.18 A	1.6	1	Double
DLV11-F	M8028	Asynchronous line interface	1.0 A	0.18 A	2.2	1	Double
DLV11-J	M8043	4 asynchronous serial interfaces	1.0 A	0.25 A	1	1	Double
DPV11	M8020	Synchronous serial line interface	1.2 A	0.30 A	1.0	1.0	Double

\* These ac load figures were measured using standard TDR (time domain reflectometry) techniques. The conversion factor is 9.35 pF/ac load. These numbers are nominal values which will tend to vary from module to module due to normal tolerances of components used in the manufacturing of the product.

Table 1-2 (cont') Module Specifications

Option Desig.	Module No(s).	Description	Power Requirements		Bus Loads*		
			+5V ±5%	+12V ±3%	AC(Max)	DC	Size
DRV11	M7941	Parallel line unit interface	0.9 A	—	1.4	1	Double
DRV11-B	M7950	DMA interface	1.9 A	—	3.3	1	Quad
DRV11-J	M8049	64-line parallel I/O	1.6 A	1.8 A	2.0	1	Double
DRV11-P	M7948	Foundation module	1.0 A +user logic	—	2.1	1	Quad
DUV11	M7951	Synchronous serial line interface	0.86 A	0.32 A	1.00	1	Quad
DZV11	M7957	Asynchronous line interface	1.15 A	0.39 A	3.95	1	Quad
FPF11	M8188	Floating point processor	5.5 A	—			Quad
H9270		4 X 4 backplane			5.1	0	
H9273		4 X 9 backplane			2.6	0	

\* These ac load figures were measured using standard TDR (time domain reflectometry) techniques. The conversion factor is 9.35 pF/ac load. These numbers are nominal values which will tend to vary from module to module due to normal tolerances of components used in the manufacturing of the product.



Table 1-2 (con't) Module Specifications

Option Desig.	Module No(s).	Description	Power Requirements		Bus Loads*		Size
			+ 5V ±5%	+ 12V ±3%	AC(Max)	DC	
H9275-A		4 X 9 backplane	—	—	10.0	0	
H9281A		2 X 4 backplane			1.3	0	
H9276		4 X 9 backplane			2.6	0	
H9281B		2 X 8 backplane			2.4	0	
H9281C		2 X 12 backplane			3.6	0	
IBV11-A	M7954	Instrument bus interface	0.8 A	—	1.9	1	Double
KD11-F	M7264	LSI-11 CPU with 4K RAM	1.8 A	0.8 A	2.4	1	Quad
KD11-H	M7264-YA	LSI-11 CPU without RAM	1.6 A	0.25 A	2.4	1	Quad
KD11-HA	M7270	LSI-11/2 CPU	1.0 A	0.22 A	1.7	1	Double
KDF-11	M8186	LSI-11/23 CPU	2.0 A	0.2 A	2.0	1	Double

\* These ac load figures were measured using standard TDR (time domain reflectometry) techniques. The conversion factor is 9.35 pF/ac load. These numbers are nominal values which will tend to vary from module to module due to normal tolerances of components used in the manufacturing of the product.

Table 1-2 (cont) Module Specifications

Option Desig.	Module No(s).	Description	Power Requirements		Bus Loads*		Size
			+ 5V ±5%	+ 12V ±3%	AC(Max)	DC	
KDF-11B	M8189	PDP-11/23-PLUS CPU with bootstrap, LTC, and two SLUS	4.5 A	.3 A	2.0	1.0	Quad
KUV-11	M8018	WCS module	3.0 A	—	—	1	Quad
KPV11-A	M8016	Power-fail/line- time clock	0.56 A	—	1.63	1	Double
KPV11-B	M8016-YB	Power-fail/line- time clock/120 Ω bus terminator	.56 A	—	1.63	1	Double
KPV11-C	M8016-YC	Power-fail/line- time clock/220 Ω bus terminator	0.56 A	—	1.63	1	Double
KWV11-A	M7952	Programmable real-time clock	1.75 A	0.01 A	3.4	1	Quad

\* These ac load figures were measured using standard TDR (time domain reflectometry) techniques. The conversion factor is 9.35 pF/ac load. These numbers are nominal values which will tend to vary from module to module due to normal tolerances of components used in the manufacturing of the product.

Table 1-2 (con't) Module Specifications

Option Desig.	Module No(s).	Description	Power Requirements		Bus Loads*		Size
			+ 5V ±5%	+ 12V ±3%	AC(Max)	DC	
KWV11-C	A4002	Programmable real-time clock	1.75 A	0.1 A	1.0	1.0	Double
KXT11-AA	M8063-AA	Single board computer	2.8 A	1.10 A	1.7	1	Double
LAV11	M7949	LA180 line printer interface	0.8 A	—	1.8	1	Double
LPV11	M8027	LA180/LP05 printer interface	0.8 A	—	1.4	1	Double
MCV11-D	M8631	8K X 32KB CMOS read/write memory	1.20 A	—	2.0	1.0	Dual
MRV11-AA	M7942	4K X 16 read-only memory (less PROM integrated circuits)	0.4 AA	—	1.8	1	Double

\* These ac load figures were measured using standard TDR (time domain reflectometry) techniques. The conversion factor is 9.35 pF/ac load. These numbers are nominal values which will tend to vary from module to module due to normal tolerances of components used in the manufacturing of the product.

Table 1-2 (con't) Module Specifications

Option Desig.	Module No(s).	Description	Power Requirements		Bus Loads*		Size
			+ 5V ±5%	+ 12V ±3%	AC(Max)	DC	
RKV11-D	M7269	LSI-11 Bus control for RKV11-D	1.8 A	—	1.9	1	Double
RLV11	M8013 M8014	RL01 disk drive	6.5 A	1.0 A	3.2	1	2 Quads
RLV12	M8061	Disk controller	5.0 A	0.1 A	3.0	1.0	Quad
RXV11	M7946	RX01 interface	1.5 A	—	1.8	1	Double
RXV21	N8029	Double density floppy interface	1.1 A	—	2.0	1	Double
TEV11	M9400-YB	120 $\Omega$ terminator	0.5 A	—	0	0	Double
TU58		Serial/cartridge cassette	0.75 A Appr.	1.2 A max			
VK170	CAM7142	Serial video module	1.2 A	0.15			Double

\* These ac load figures were measured using standard TDR (time domain reflectometry) techniques. The conversion factor is 9.35 pF/ac load. These numbers are nominal values which will tend to vary from module to module due to normal tolerances of components used in the manufacturing of the product.





## CHAPTER 2

# ARCHITECTURAL OVERVIEW

### SYSTEM ARCHITECTURE

A complete and powerful microcomputer system can be configured using an LSI-11 microcomputer, appropriate memory, I/O devices, and interconnection hardware. DIGITAL's industry standard LSI-11 Bus provides communication between system components. A typical system configuration is shown in Figure 2-1.

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, 4 extended address lines, 6 data transfer control lines, 6 system control lines, and 10 interrupt and Direct Memory Access (DMA) control lines.

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 use of common bus interfaces for different devices and simplifies the interface design.

Communication between two devices on the bus is accomplished by 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, 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.

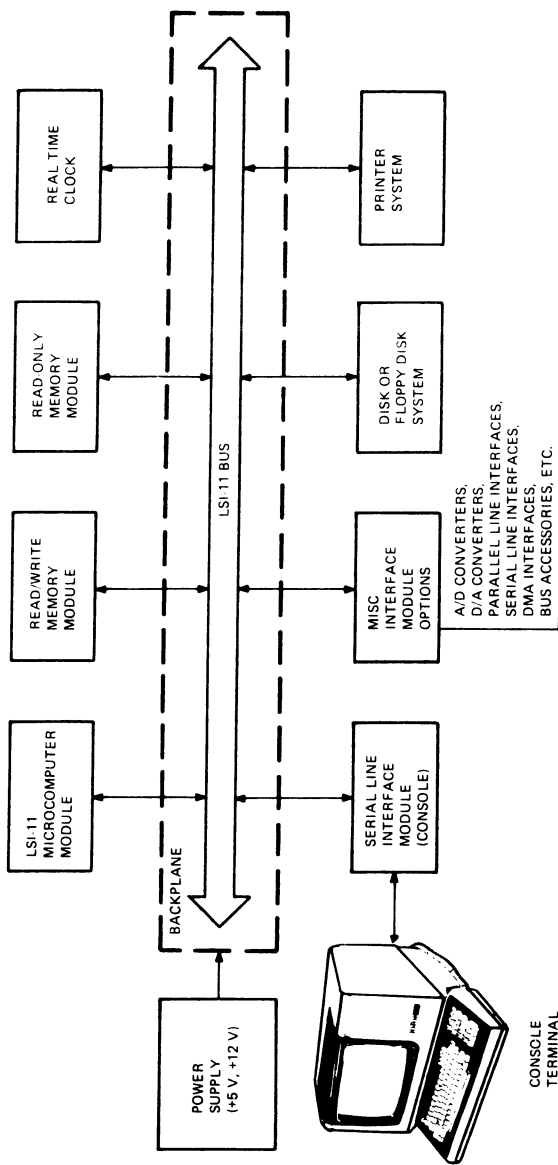


Figure 2-1 Typical LSI-11 System Configuration



The microcomputer module controls the time allocation of the LSI-11 Bus for peripherals, and performs arithmetic and logic operations, as well as instruction decoding.

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 I/O device, and permits DMA transfers directly between I/O devices and memory without disturbing the processor registers. 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.

### **General-Purpose Registers**

The LSI-11 central processor module contains eight 16-bit general-purpose registers that perform a variety of functions. These registers serve as accumulators, index registers, autoincrement registers, autodecrement registers, or as stack pointers for temporary storage of data. Arithmetic operations can transfer from one general-purpose register to another, from one memory location or device register to another, or between memory locations, or a device register and a general-purpose register. The eight 16-bit general-purpose registers, R0 through R7, are identified in Figure 2-2.

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 require bus cycles and longer execution time. Thus, general-purpose registers used for processor operations result in faster execution times. The bus cycles required for memory and device references are listed below.

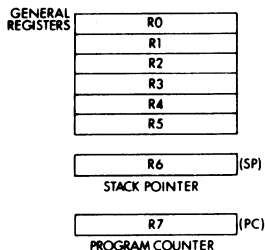


Figure 2-2 General Register Identification

**Bus Cycles**

The processor bus cycles 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
DATO(B)	Data word (Byte) transfer output	Equivalent to a Write/Word (Byte) operation

**Addressing Memory and Peripherals**

The maximum direct address space of the FALCON SBC-11/21 and LSI-11/2 is 64 KB. For the LSI-11/23, the maximum direct address space is 4 megabytes. LSI-11 memory locations and peripheral device registers are addressed in the same manner. The upper 8 KB of address space are reserved (by PDP-11 convention) for peripheral device addressing.

An LSI-11 word is divided into a high byte and a low byte, as illustrated in Figure 2-3.

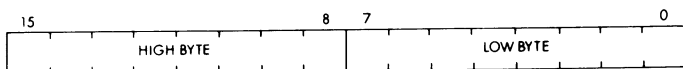


Figure 2-3 High and Low Byte

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 Figure 2-4.

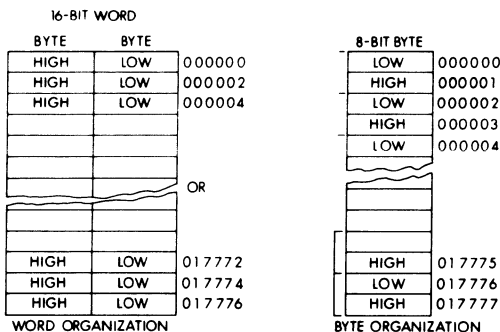


Figure 2-4 Word and Byte Addresses for First 4K Bank

Certain memory locations have been reserved by convention for interrupt and trap handling and for peripheral device registers. Addresses from 0 to 376<sub>8</sub> are reserved for trap and device interrupt vector locations. Several of these are reserved in particular for system (processor initiated) traps.

### LSI-11 Memory Organization

LSI-11 memory organization is shown in Figure 2-5.

### PROCESSOR STATUS WORD

The Processor Status Word (PSW or 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 PSW word format is shown in Figure 2-6. Certain instructions allow programmed manipulation of condition code bits and loading or storing (moving) the PSW.

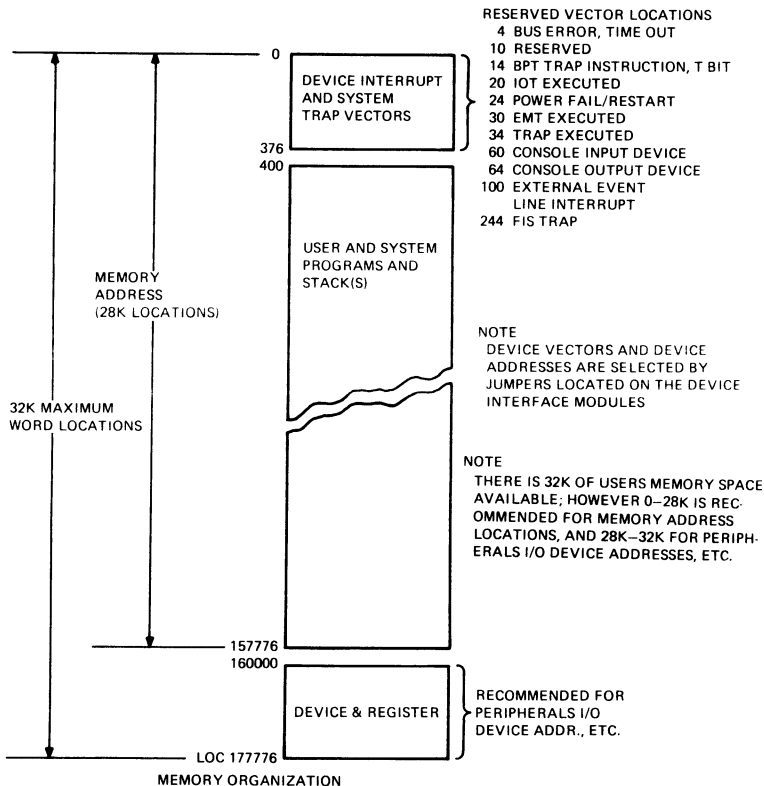


Figure 2-5 Memory Organization

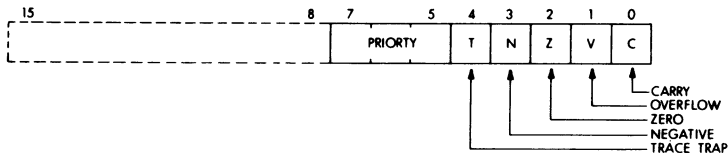


Figure 2-6 Processor Status Word (PSW)

### Interrupt Priority Bit

The processor operates with PSW bits <5:7> determining the effective processor priority, from 0 to 7. To be recognized, the external interrupt must have a higher priority than the processor. As compared to other PDP-11s, the LSI-11/2 services interrupts at one priority level, level 4. The LSI-11/23 and FALCON SBC-11/21 are capable of servicing interrupts at four levels of priority. The functions contained in the high byte pertain only to the LSI-11/23.

### 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 set or clear the trap (T bit) only 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

DIGITAL's PDP-11 instruction set offers you the opportunity to take advantage of more than a decade of PDP-11 family development and experience.

This instruction set uses the flexibility of the general-purpose registers to provide more than 87 PDP-11 standard instruction operations—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, operator or accumulator control instructions, and I/O instructions), all LSI-11 data manipulation operations are accomplished with one set of instructions. Instructions that 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 disturb-

ing the general registers. Programs can add or compare data either logically or arithmetically in a device register.

### **Addressing Modes**

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.

## **THE FALCON SBC-11/21 ARCHITECTURAL AND OPERATIONAL FEATURES**

### **The MPU**

The FALCON SBC-11/21 is based on a new 40-pin PDP-11 microprocessor. This newly developed single chip executes the PDP-11 base-level instruction set and is capable of addressing 64 KB of memory. Since the MPU has a bounded microcode design, it does not support any optional PDP-11 instructions. This one chip preserves the architecture and instruction set of the LSI-11 and LSI-11/2 while allowing the FALCON to operate at twice the speed of the LSI-11/2.

### **Optional Features**

The FALCON SBC-11/21 does not support optional PDP-11 instructions, but does have functionality never before offered on a double-height board, such as: 4 KB of RAM memory, four 28-pin sockets for RAM/ROM memory, two asynchronous serial I/O ports, 16-line bidirectional parallel I/O port, and a real-time clock.

## **THE LSI-11/23 ARCHITECTURAL AND OPERATIONAL FEATURES**

DIGITAL's LSI-11/23 offers peak performance at the board level. The LSI-11/23 retains the same architectural features as its predecessor, the LSI-11/2, and has design enhancements which permit greater execution speeds, multilevel interrupts, full memory management, and the PDP-11/34 floating point instruction set. The LSI-11/23 represents state-of-the-art N-MOS technology and printed circuit design.

### **The Chip Set**

The major architectural and functional differences between the LSI-11/2 and the LSI-11/23 are inherent in the latter's basic chip set design. Its data and control chips are mounted on one dual carrier pack-

age. A 16-bit Microinstruction Bus and a 16-bit Data Address Bus provide the communication between chips, and between chips and the LSI-11 Bus.

The data chip maintains the same basic features as the earlier LSI-11/2. It contains the arithmetic logic unit, the register file, the data access ports, and the processor status word. The major difference in the structure of the LSI-11/23 data chip is the wider data path that supports a more powerful microcode for greater overall performance.

The LSI-11/23 control chip is functionally similar to the LSI-11/2 control and MICROM units. The combined functions of these units have been implemented in one silicon gate MOS chip. The LSI-11/23 control chip contains microprogram storage, contained in both ROM and programmable logic arrays.

Two chips on one carrier provide an efficient way to transfer the LSI-11/23 powerful microcode to and from the internal logic circuits of the data and control chips. This innovation in Large Scale Integration not only shortens the interconnected path between the chips, but consequently provides efficient utilization of CPU cycle time by shortening internal microcoded logical sequences which result in increased execution speeds. This unique chip set organization also allows room for EIS as standard.

### **Optional Features**

**Memory Management Unit** — The Memory Management Unit (MMU) allows memory addressing beyond 64 KB up to 4 megabytes. As part of the LSI-11/23 chip set, the MMU enables the processor to operate in either kernel or user mode. In kernel mode, the operating system and programs have complete control and execute all instructions. In user mode, programs are prohibited from performing instructions that could modify the kernel program, halt the computer, or access memory space reserved for the kernel or other users.

The Memory Management Unit provides address relocation to extend the physical address range to 22 bits and incorporates unique features of (1) segmentation, e.g., dividing large segments of program memory into smaller, more efficient segments and (2) protection, the ability to control and restrict access to a memory segment by an unauthorized user. These features are not available in other microcomputers.

**Floating Point Option** — optional Floating Point units are available for the LSI-11/23. These units implement 46 microcoded instructions that perform arithmetic, logical, and conversion operations, and operate five to ten times faster than equivalent software routines. The KEF11 and FPF11 Floating Point Option instructions are completely compati-

ble with the PDP-11/34 (FP11-A) Floating Point Processor Instruction Set.

The Memory Management Unit is required when using the KEF11, because it contains eight 64-bit floating point registers. Instructions operate with 32-bit single-precision and 64-bit double-precision data, equivalent to 7 and 17 decimal digit accuracy, respectively.

## **THE LSI-11/2 ARCHITECTURAL AND OPERATIONAL FEATURES**

### **The Chip Set**

The heart of the double-height LSI-11/2 is a set of N-channel MOS chips containing thousands of logical circuits. Interchip communication is accomplished by a 22-bit internal Microinstruction Bus and 16-bit Data Address Lines. These features permit internal control of the logical circuits of the board, transmission of data between the MOS chips, and communication between the processor and the LSI-11 Bus.

Four 40-pin packages house a data chip, a control chip, and two ROM (MICROM) chips containing microcode. The data chip contains the register file, the arithmetic logic unit, the processor status word, and the data access ports. The control chip decodes instructions, handles interrupts, and directs the flow of information.

The two MICROM chips contain microcode to implement the basic PDP-11 instruction set as well as the ODT/ASCII console routine. An optional third MICROM chip contains microcode to implement the EIS/FIS integer and floating point instructions.

### **Optional Features**

For number-crunching FORTRAN computations, the LSI-11/2 offers both fixed and floating point instructions implemented with one chip mounted on a 40-pin carrier. The chip implements four floating point instructions called the Floating Point Instruction Set (FIS). These instructions offer floating addition, subtraction, multiplication, and division, with the added capability of enabling direct operations on single-precision 32-bit operands. Also resident on the chip is the Extended Instruction Set (EIS) that offers hardware integer multiply and divide as well as allowing the direct implementation of multiple shifts.

### **LEVELS OF INTEGRATION**

The LSI-11 family of microcomputers is available in three levels of integration: boards, boxes, and systems. This choice of three integration levels allows customers to purchase any optimal level of integration to suit their specific need or requirement.



## **Boards**

The LSI-11 Bus concept permits modules to be chosen and assembled into a system that fills the specific needs of any customer. Boards may be assembled in a DIGITAL-supplied mounting chassis or in a backplane powered and cooled by the user. This latter alternative offers wide packaging flexibility for the user. Table 1-2 (in Chapter 1 of this Handbook) summarizes available modules and their operational specifications.

All card guide and backplane assemblies contain the LSI-11 Bus in a convenient package. These card guide assemblies support the broadest choice of high-performance options in the industry.

There are six versions of the card guide mounting chassis. Three versions, the H9281 series, available in sizes  $2 \times 4$ ,  $2 \times 8$ , and  $2 \times 12$ , support double-height boards, and accommodate 4, 8, or 12 modules in a variety of configurations. The H9281 card cages that support 8 and 12 double-height family modules also include built-in bus termination.

Three card cages are available in the H9270 series in sizes of  $4 \times 4$  and  $4 \times 9$ . These card cages support both quad and double-height modules.

The newest card cages in the H9270 series are the H9275-A and the H9276-A. The H9275-A is a  $4 \times 9$  inch backplane that supports 22 address lines, and will accept up to 18 double-height or quad-height modules. Bus termination is built into this backplane. The H9276-A is a  $4 \times 9$  backplane that will accept 9 double- or quad-height modules. This backplane may be cabled or expanded to other H9276 backplanes to allow for larger configurations.

In order to address the broadest customer need, a seventh backplane, the DDV11-B (without card guides) is available. This backplane accommodates 18 double-height or 9 quad-height modules, and provides two unbusied slots, E and F, for customer-designed modules.

## **Mounting Boxes**

For ultimate design flexibility and faster time to market, DIGITAL offers three mounting enclosures, including power supply, cooling fans, and backplane.

For compact, low-cost systems, DIGITAL offers the BA11-VA mounting box. This chassis houses a  $2 \times 4$  LSI-11 backplane, as well as a power supply and cooling fan.

The BA11-ME expansion enclosure houses a  $4 \times 4$  backplane and utilizes the H780 power supply.

The third expansion enclosure is the BA11-NE. This chassis houses a 4 x 9 backplane which accepts up to 9 double-height or 9 quad-height modules.

### **Microcomputer Boxes**

Microcomputer boxes offer an intermediate level of integration that consists of a properly matched combination of mounting box, CPU, memory, and I/O controllers. These boxes provide an engine for customer hardware tailored for industrial, technical, and scientific laboratories.

### **Systems**

Systems, being the highest level of integration, provide the most complete solution. Systems products include the basic computer plus mass storage, console terminal, and operating system software.

A special case is the packaged development system used for development of board-level target systems, described below.

### **Packaged Development Systems**

Packaged development systems can be obtained from DIGITAL, offering complete hardware and software products that provide everything you need to develop a computerized solution for your microcomputer application. The LSI-11 MDS development system family provides software debugging and development tools which are tailored to microcomputer applications. These development systems can be installed by the customer and are designed for ease of installation. The user documentation is designed to simplify the development process for the first-time user.

### **The PB11 PROM Programmer**

To complete the necessary tools to develop your PROM-based application, Digital offers the PB11. This option supports a variety of both PROM and EPROM chips. It consists of a software utility which runs on any RT-11-based system and a programmable PROM burner which is connected to your development system via a serial link. The software utility allows the user to program PROMs using easy-to-understand commands from a terminal.





## CHAPTER 3

# ADDRESSING MODES

In the LSI-11 and PDP-11 families, memory reference addressing is accomplished using the eight general-purpose registers. To specify the location of data (operand address), one of the eight registers is selected with an accompanying addressing mode. Each memory reference instruction specifies the:

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

Together, the instruction set format and addressing modes available to the programmer are of particular importance. This combination allows the user to take advantage of LSI-11 family benefits and 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 capability allows easy access to variable entries in a list.

Using registers for both data manipulation and address calculation result 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**. See the Programming Techniques Chapter for a discussion about using the stack. Register 6 is always used as the hardware stack pointer (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. See the Instruction Set Chapter for an explanation of the full instruction set and instruction formats.

Table 3-1 contains a listing and description of the instructions used in the examples (within this chapter) to illustrate the power and flexibility of the PDP-11 addressing modes.

**Table 3-1 Example Instructions**

<b>Mnemonic</b>	<b>Description</b>	<b>Octal Code</b>
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)

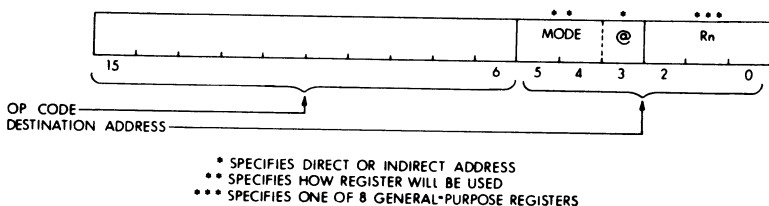
SS = source field (6 bits)

() = contents of

Single- and double-operand instructions utilize the following formats.

The instruction format for all single-operand instructions (such as Clear, Increment, Test) is:

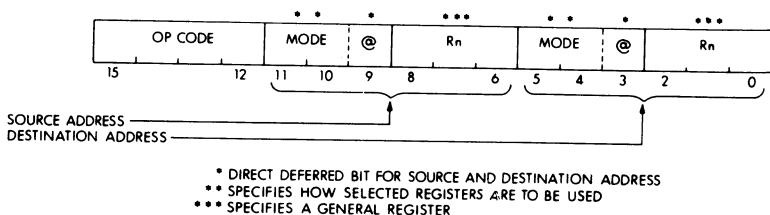
## Chapter 3 — Addressing Modes



### Single-Operand Instruction Format

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

The instruction format of the double-operand instruction is:



### Double-Operand Instruction Format

Bits 3-5 and 9-11 specify the binary code of the addressing modes 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-purpose registers can be used as simple accumulators. The operand is contained in the selected register. Assembler syntax requires that a general-purpose register be defined as follows:

R0 = %0

R1 = %1

R2 = %2

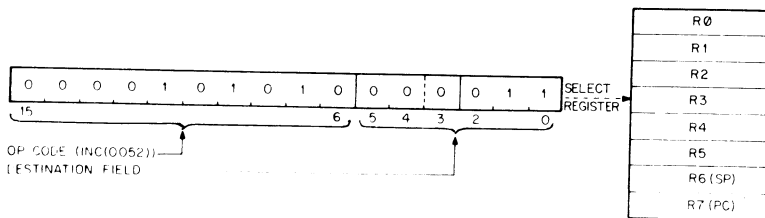
% sign indicates register definition.

### Register Mode Example

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



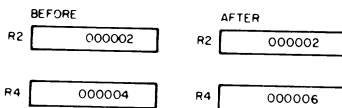
Represented as:



### 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)

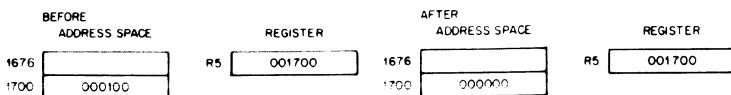
In register deferred mode, the address of the operand is stored in a general-purpose register. The address contained in the general-purpose register directs the CPU to the operand. The operand is located outside the CPU, either in memory, or in an I/O register.

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:

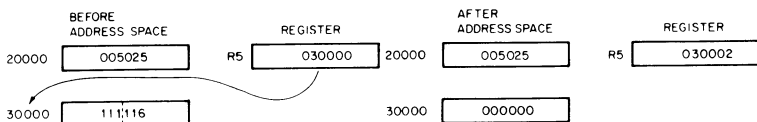
**AUTOINCREMENT MODE****MODE 2 (Rn)+**

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:



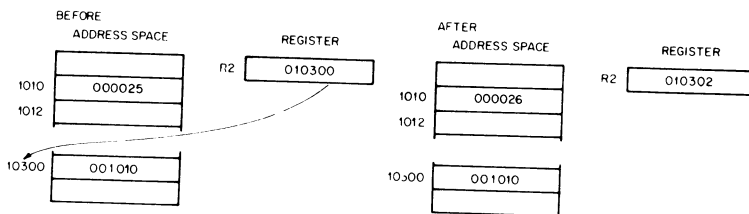
## AUTOINCREMENT DEFERRED MODE MODE 3    @(Rn)+

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 data, 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:



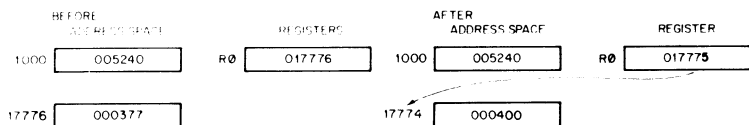
## AUTODECREMENT MODE MODE 4    -(Rn)

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:

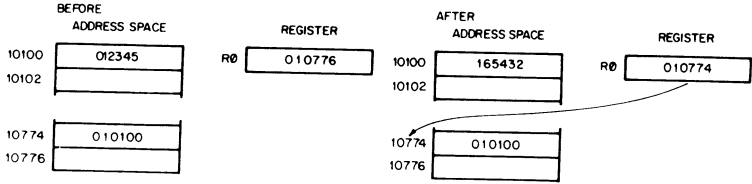
**AUTODECREMENT DEFERRED MODE      MODE 5      @ $-(Rn)$** 

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**

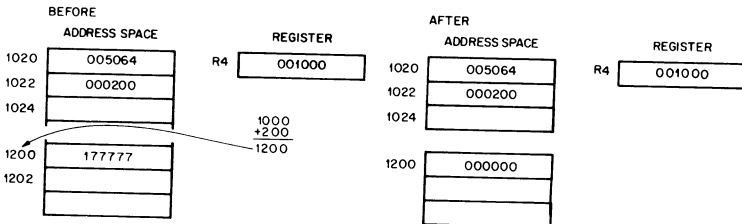
**X(Rn)**

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:



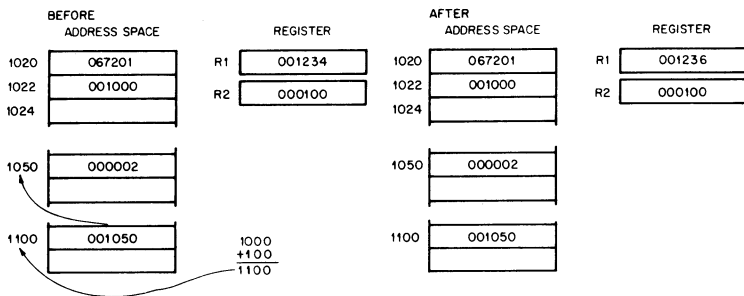
**INDEX DEFERRED MODE****MODE 7****@X(Rn)**

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**

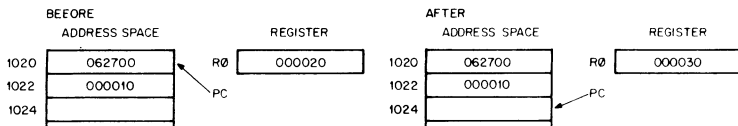
**#n**

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 instruction) 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**

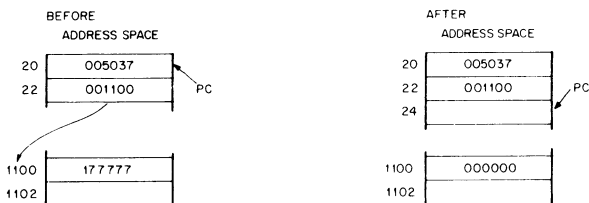
**Description**

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



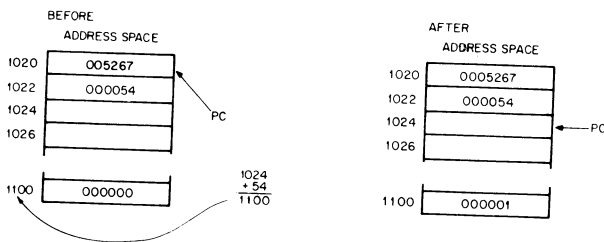
always determined with respect to the updated PC. Therefore, when the instruction is relocated, the operand remains the same relative distance away.

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

Represented as:

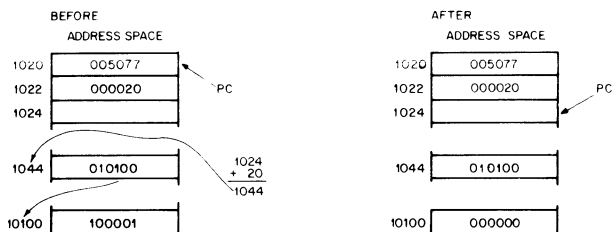


Table 3-2 summarizes the direct addressing modes.

**Table 3-2 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.

Table 3-2 (con'd) Direct Addressing Modes

Binary Code	Mode	Name	Symbolic	Function
110	6	Index	X(Rn)	Value X is added to (Rn) to produce address of operand. Neither X nor (Rn) is modified.

Table 3-3 summarizes the indirect addressing modes.

Table 3-3 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, <b>even</b> for byte instructions).
101	5	Autodecrement Deferred	@-(Rn)	Register is decremented (always by 2, <b>even</b> for byte instructions) and then used as a pointer to a word containing the address of the operand.

**Table 3-3 (con'd) Indirect Addressing Modes**

<b>Binary Code</b>	<b>Mode</b>	<b>Name</b>	<b>Symbolic</b>	<b>Function</b>
111	7	Index De-ferred	@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 immediate, absolute (or immediate deferred), relative, and relative deferred.

Table 3-4 summarizes the PC register addressing modes.

**Table 3-4 PC Register Addressing Modes**

<b>Binary Code</b>	<b>Mode</b>	<b>Name</b>	<b>Symbolic</b>	<b>Function</b>
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.

**Table 3-4 (con'd) PC Register Addressing Modes**

Binary Code	Mode	Name	Symbolic	Function
111	7	Relative De-ferred	@A	Address of A, relative to the instruction, is contained in the instruction. Operand address is contained in A.

**GRAPHIC SUMMARY OF PDP-11 ADDRESSING MODES**

**General Register Addressing Modes**

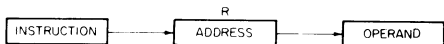
R is a general register, 0 to 7.

(R) is the contents of that register.

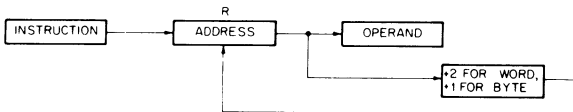
**Mode 0 Register** OPR R R contains operand.



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

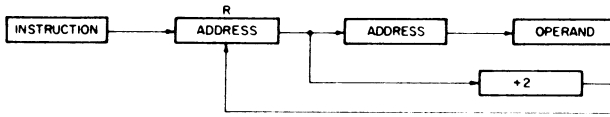


**Mode 2 Autoincrement** OPR (R)+ R contains address, then increment (R).

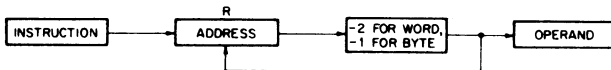


## Chapter 3 — Addressing Modes

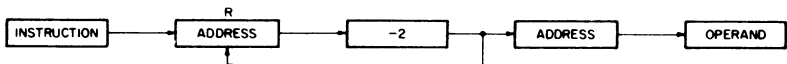
**Mode 3      Autoincrement deferred**      OPR  $@(R)+$       R contains address of address, then increment (R) by 2.



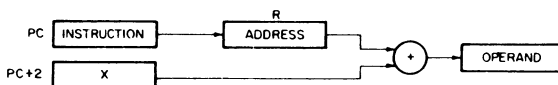
**Mode 4      Autodecrement**      OPR  $-(R)$       Decrement (R), then R contains address.



**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**

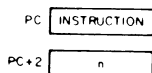
Register = 7

**Mode 2**

**Immediate**

OPR #n

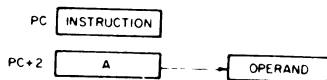
Literal operand n  
is contained in  
the instruction.



**Mode 3 Absolute**

OPR @#A

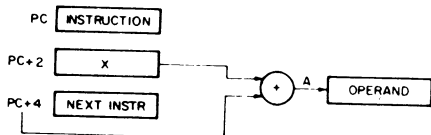
Address A is  
contained in the  
instruction.



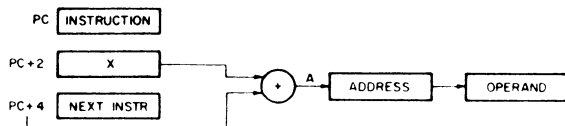
**Mode 6 Relative**

OPR A

PC+4 + X is ad-  
dress. PC+4 is  
updated PC.

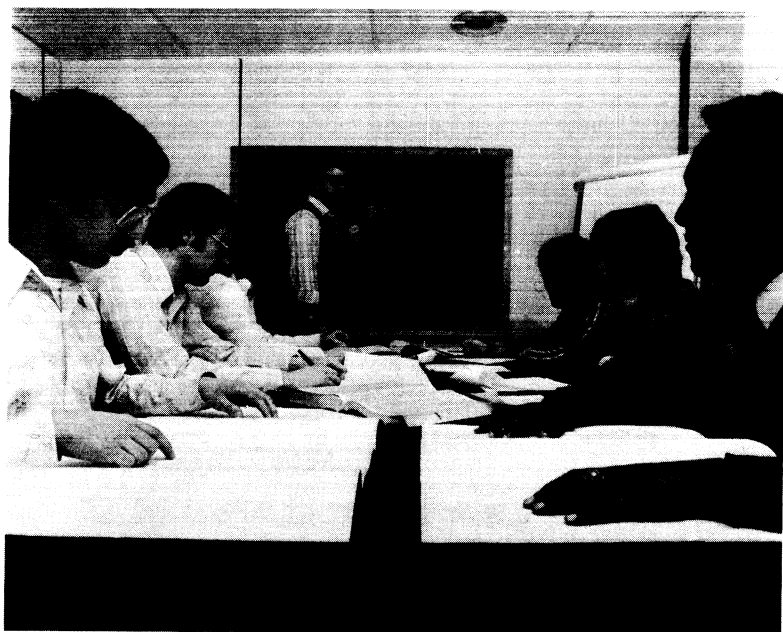


**Mode 7**      **Relative deferred**      **OPR @A**      PC+4 + X is address of address.  
PC+4 is updated PC.









## CHAPTER 4

# INSTRUCTION SET

The LSI-11 instruction set offers a wide choice of operations and addressing modes. To save memory space and to simplify the implementation of control and communications applications, the instructions allow byte and word addressing in both single- and double-operand formats. Using double-operand instructions, you can 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:

```
LOAD A
ADD B
STORE B
```

The instruction set also contains a full set of conditional branches to eliminate excessive use of jump instructions.

Functionally, the instructions fall into the following categories:

- *Single-Operand* — The first part of the word, called the “op code,” specifies the operation; the second part provides information for locating the operand. Figure 4-1 illustrates the single-operand instruction format.
- *Double-Operand* — The first part of the word specifies the operation to be performed; the remaining two parts provide information for locating two operands. Figure 4-2 illustrates the double-operand instruction format.
- *Branch* — 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. Figure 4-3 illustrates the branch instruction format.
- *Jump and Subroutine* — These instructions have both op code and address parts, and, in the case of JSR, a register for linkage. Figure 4-4 illustrates the JSR instruction format and Figure 4-5 illustrates the RTS instruction format.
- *Trap* — These instructions contain an op code only. In TRAP and EMT, the low-order byte may be used for function dispatching.
- *Miscellaneous* — HALT, WAIT, and Memory Management.
- *Condition Code* — These instructions set or clear the condition codes. Figure 4-6 illustrates the condition code format.

**SINGLE-OPERAND INSTRUCTIONS**

	<b>Mnemonic</b>	<b>Instruction</b>
<b>General</b>	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
	NOP	no operation
<b>Shift &amp; Rotate</b>	ASR(B)	arithmetic shift right
	ASL(B)	arithmetic shift left
	ROR(B)	rotate right
	ROL(B)	rotate left
	SWAB	swap bytes
<b>Multiple Precision</b>	ADC(B)	add carry
	SBC(B)	subtract carry
	SXT	sign extend
	MFPS	move byte from processor status
	MTPS	move byte to processor status

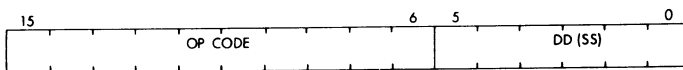
**Instruction Format**

Figure 4-1 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**

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

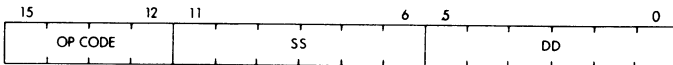
**Instruction Format**

Figure 4-2 Double-Operand Instruction Format

The format of most double-operand instructions, though similar to that of single-operand instructions, has *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 from the mode and register used by another field.

- Bit 15 indicates word or byte operation *except* when used with op code 6, in which case 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.

- Some double-operand instructions (ASH, ASHC, MUL, DIV) must have the destination operand only in a register. Bits 15-9 specify the op code. Bits 8-6 specify the destination register. Bits 5-0 contain the source field. XOR has a similar format, except that the source is in a register specified by bits 8-6, and the destination field is specified by bits 5-0. As part of the EIS option, these instructions are standard on the LSI-11/23, PDP-11/23, and the PDP-11/23-PLUS, and optional on the LSI-11, LSI-11/2, and the PDP-11/03.

### 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 MOV<sub>B</sub>. There are no byte operations for ADD and SUB, i.e., no ADD<sub>B</sub> or SUB<sub>B</sub>.

## PROGRAM CONTROL INSTRUCTIONS

### Branch Instructions

Branch	Mnemonic	Instruction
	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

### 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

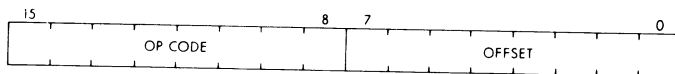
**Instruction Format**

Figure 4-3 Branch Instruction Format

- The high byte (bits 15-8) of the instruction is an op code specifying the conditions to be tested.
- The low byte (bits 7-0) 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
JMP	jump
JSR	jump to subroutine
RTS	return from subroutine

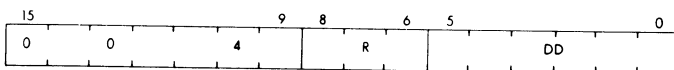
**Instruction Format****JSR Format**

Figure 4-4 JSR Instruction Format

- Bits 15-9 are always octal 004, the op code for JSR.
- Bits 8-6 specify the link register. Any general-purpose register may be used in the link, except R6.
- Bits 5-0 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.

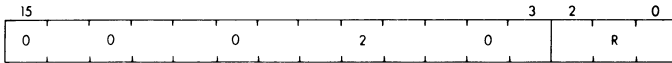
**RTS Format**

Figure 4-5 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 15-3 always contain octal 00020, which is the op code for RTS.
- Bits 2-0 specify any one of the general-purpose registers.
- The register specified by bits 2-0 must be the same register used as the link between the JSR causing the jump and the RTS returning control.

**TRAPS AND INTERRUPTS**

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

The three ways to leave a main program are:

1. *Software exit* — the program specifies a jump to some subroutine.
2. *Trap exit* — internal hardware on a special instruction forces a jump to an error handling routine.
3. *Interrupt exit* — external hardware forces a jump to an interrupt service routine.

In each case, a jump to another program takes place. Once the latter program has been executed, control is returned to the proper point in the main program.

**MISCELLANEOUS INSTRUCTIONS**

<b>Mnemonic</b>	<b>Instruction</b>
HALT	halt
WAIT	wait for interrupt
RESET	reset bus
MTPD	move to previous data space



**Mnemonic    Instruction**

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
MFPT	move from processor type

**CONDITION-CODE OPERATION**

<b>Mnemonic</b>	<b>Instruction</b>
CLC, CLV, CLZ, CLN, CCC	clear
SEC, SEV, SEZ, SEN, SCC	set

The four condition-code bits are:

- 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. The hardware uses one of two methods to check for an overflow condition:

1. 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.
2. 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.

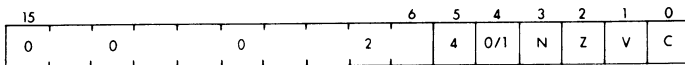


Figure 4-6 Condition-Code Operators' Format

### Instruction Format

The format of the condition code operators is:

- 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<sub>8</sub> byte locations beginning at memory address 600.

R0 = 600

R1 = 30

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

### Program Description

- The CLRB (R0)+ instruction clears the content of the location specified by R0 and increments R0.
- R0 is the pointer.
- Because the autoincrement 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 come 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 have 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 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 the 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.

### 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.

<b>SYMBOL</b>	<b>MEANING</b>
MN	Maintenance Instruction
SO	Single-Operand Instruction
DO	Double-Operand Instruction
PC	Program Control Instruction
MS	Miscellaneous Instruction
CC	Condition Code
(X)	Contents of memory location whose address is X. For example, (R5) means the contents of the memory location whose address is contained in R5.
src	Source Address
dst	Destination Address
tmp	Contents of temporary internal register
←	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
	Logical AND
v	Logical inclusive OR (either one or both)
∨	Logical exclusive OR (either one, but not both)
~	Logical NOT
Reg or R	Contents of Register
B	Byte
MPI	Most Positive Integer—077777 (word) or 177 (byte)
MNI	Most Negative Integer—100000 (word) or 200 (byte)

**Summary of Basic Instruction Set****INSTRUCTION SET**

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

The Extended Instruction Set (EIS) is standard on the LSI-11/23, PDP-11/23, and the PDP-11/23-PLUS. It is available as an option on the LSI-11, LSI-11/2, and the PDP-11/03, and can be referenced at the end of this chapter. Some members of the PDP-11 family have slight differences in the way instructions are executed. Refer to Appendix D for detailed family differences.

ADC	COM
ADCB	COMB
ADD	DEC
ASL	DECB
ASLB	EMT
ASR	HALT
ASRB	INC
BCC	INCB
BCS	IOT
BEQ	JMP
BGT	JSR
BHI	MARK
BHIS	MOV
BIC	MOVB
BICB	NEG
BIS	NEGB
BISB	NOP
BIT	RESET
BITB	ROL
BLE	ROLB
BLO	ROR
BLOS	RORB
BLT	RTI
BMI	RTS
BNE	RTT
BPL	SBC
BPT	SBCB
BR	SCC, SEN, SEZ, SEV, SEC
BVC	SOB
BVS	SUB, SXT
CLR	SWAB

CLRB	TRAP
CCC, CLN, CLZ, CLV, CLC	TST
CMP	TSTB
CMPR	XOR
	WAIT

These basic instructions are standard on:

- LSI-11
- LSI-11/2
- LSI-11/23
- FALCON SBC-11/21 (except for MARK instruction which is not implemented on FALCON SBC-11/21.)
- PDP-11/03
- PDP-11/23
- PDP-11/23-PLUS

**MFPD, MTPD, MTPI, MFPI**

Available on the LSI-11/23, PDP-11/23, and PDP-11/23-PLUS only.

**MFPS, MTPS**

Available on the LSI-11, LSI-11/2, LSI-11/23, FALCON SBC-11/21, PDP-11/03, PDP-11/23, and PDP-11/23-PLUS.

**MFPT**

Available on the FALCON SBC-11/21, LSI-11/23, PDP-11/23, and the PDP-11/23-PLUS.

**ADC  
ADCB**

Add Carry	SO	0055DD 1055DD
-----------	----	------------------

**Operation:** (dst) ← (dst)+C

**Condition** N: set if result < 0

**Codes:** Z: set if result = 0

V: set if (dst) is M.P.I. and C = 1

C: set if (dst) is -1 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:**  $(dst) \leftarrow (src) + (dst)$

**Condition** N: set if result < 0

**Codes:** Z: set if result = 0

V: set if there is arithmetic overflow as a result of the operation; that is, both operands were of the same sign and the result is of the opposite sign

C: set if there is a carry from the most significant bit of the result

**Description:** Adds the source operand to the destination operand and stores the result at the destination address. The original contents of the destination are lost. The contents of the source are not affected. Two's complement addition is performed.

**ASL  
ASLB**

Arithmetic Shift Left	SO	0063DD
	SO	1063DD

**Operation:**  $(dst) \leftarrow (dst)$  shifted one place to the left

**Condition** N: set if high-order bit of the result is set (result < 0)

**Codes:** Z: set if the result = 0

V: loaded with the exclusive OR of the N bit and C bit (as set by the completion of the shift operation)

C: loaded with the high-order bit of the destination

**Description:** Shifts all bits of the destination left one place. The low-order bit is loaded with a 0. The C bit of the status word is loaded from the high-order bit of the destination. ASL performs a signed multiplication of the destination by 2 with overflow indication. For example, -1 shifted left yields -2, +2 shifted left yields +4, and -3 shifted left yields -6.

**ASR  
ASRB**

Arithmetic Shift	SO	0062DD
Right	SO	1062DD

**Operation:** (dst) ← (dst) shifted one place to the right

**Condition** N: set if the high-order bit of the result is set (result < 0)

**Codes:**

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, rounded to minus infinity. -1 shifted right remains -1, +5 shifted right yields +2, -5 shifted right yields -3.

**BCC**

Branch if Carry	PC	103000
Clear		Plus 8-bit offset

**Operation:** PC ← PC + (2 × offset) if C = 0

**Condition** N: unaffected

**Codes:** 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
		Plus 8-bit offset

**Operation:**  $PC \leftarrow PC + (2 \times \text{offset})$  if  $C = 1$

**Condition** N: unaffected

**Codes:** 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 (to zero)	PC	001400 Plus 8-bit offset
------------------------------	----	--------------------------------

**Operation:**  $PC \leftarrow PC + (2 \times \text{offset})$  if  $Z = 1$

**Condition** N: unaffected

**Codes:** 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 Plus 8-bit offset
------------------------------------	----	--------------------------------

**Operation:**  $PC \leftarrow PC + (2 \times \text{offset})$  if  $N \nleftrightarrow V = 0$

**Condition** N: unaffected

**Codes:** 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 Plus 8-bit offset
------------------------	----	--------------------------------

**Operation:** PC  $\leftarrow$  PC + (2 × offset) if Z v (N  $\nabla$  V) = 0

**Condition** N: unaffected

**Codes:** Z: unaffected

V: unaffected

C: unaffected

**Description:** Causes a branch if Z is clear (result of previous operation  $\neq$  0) and N = V. Thus, BGT always branches following an operation that added two negative numbers, even if overflow occurred. 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 Plus 8-bit offset
------------------	----	--------------------------------

**Operation:** PC  $\leftarrow$  PC + (2 × offset) if C = 0 and Z = 0

**Condition** N: unaffected

**Codes:** 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 or the Same      PC      103000  
Plus 8-bit offset

**Operation:**  $PC \leftarrow PC + (2 \times \text{offset})$  if C = 0

**Condition**      N: unaffected

**Codes:**      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:**  $(\text{dst}) \leftarrow \sim(\text{src}) \wedge (\text{dst})$

**Condition**      N: set if high-order bit of result set

**Codes:**      Z: set if result = 0

V: cleared

C: not cleared

**Description:** Clears each bit in the destination that corresponds to a set bit in the source. The original contents of the destination are lost. The contents of the source are unaffected.

**BIS  
BISB**

Bit Set DO 05SSDD  
15SSDD

**Operation:** (dst) ← (src) v (dst)

**Condition** N: set if high-order bit of result set

**Codes:** Z: set if result = 0

V: cleared

C: unaffected

**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** N: set if high-order bit of result set

**Codes:** Z: set if result = 0

V: cleared

C: unaffected

**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 PC 003400  
or Equal To Plus 8-bit  
offset

**Operation:** PC ← PC + (2 × offset) if Zv (N ≠ V) = 1

**Condition** N: unaffected

**Codes:** Z: unaffected

V: unaffected

C: unaffected

**Description:** Causes a branch if Z is set or if N does not equal V. Thus, BLE always branches following an operation that added two negative numbers, even if overflow occurred. In particular, BLE always causes a branch if it follows a CMP instruction operating on a negative source and a positive destination (even if overflow occurred). Further, BLE never causes a branch when it follows a CMP instruction operating on a positive source and negative destination. BLE always causes a branch if the result of the previous operation was 0.

**BLO**

Branch if Lower                      PC                                      1034000

**Operation:**                       $PC \leftarrow PC + (2 \times \text{offset})$  if C = 1

**Condition**                      N: unaffected

**Codes:**                              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  
Plus 8-bit  
offset

**Operation:**                       $PC \leftarrow PC + (2 \times \text{offset})$  if CvZ = 1

**Condition**                      N: unaffected

**Codes:**                              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 to or has a lower unassigned value than the destination.





**BNE**

Branch if Not Equal    PC                            001000  
    Plus 8-bit  
    offset

**Operation:**             $PC \leftarrow PC + (2 \times \text{offset})$  if  $Z = 0$

**Condition**            N: unaffected

**Codes:**                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  
    Plus 8-bit  
    offset

**Operation:**             $PC \leftarrow PC + (2 \times \text{offset})$  if  $N = 0$

**Condition**            N: unaffected

**Codes:**                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:**             $-(SP) \leftarrow PS$

$-(SP) \leftarrow PC$

$PC \leftarrow (14)$

$PS \leftarrow (16)$

**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. Instruction has no operand.

**BR**

Branch (Unconditional)      PC      000400  
 Plus 8-bit offset

**Operation:**  $PC \leftarrow PC + (2 \times \text{offset})$

**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 one-word instruction. An unconditional branch.

**BVC**

Branch if V Bit Clear      PC      102000  
 Plus 8-bit offset

**Operation:**  $PC \leftarrow PC + (2 \times \text{offset})$  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  
Plus 8-bit  
offset

**Operation:**       $PC \leftarrow PC + (2 \times \text{offset})$  if  $V = 1$

**Condition**      N: unaffected

**Codes:**      Z: unaffected

V: unaffected

C: unaffected

**Description:**      Tests the state of V bit and causes a branch if the V bit is set. BVS is used to detect arithmetic overflow in the previous operation.

**CLR  
CLRB**

Clear      SO      0050DD  
1050DD

**Operation:**       $(\text{dst}) \leftarrow 0$

**Condition**      N: cleared

**Codes:**      Z: set

V: cleared

C: cleared

**Description:**      Contents of specified destination are replaced with zeros.

**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 perform a DATIO cycle for the last bus cycle as a “don’t care” for hardware minimization.

**C**

Clear Selected Con-      CC      000240  
dition Code Bits      Plus 4-  
bit mask

**Operation:** PSW <3:0> ← PSW <3:0>[~mask <3:0>]

**Description:** Clear condition code bits. Selectable combinations of these bits may be cleared together. Condition code bits corresponding to bits in the condition code operator (bits 0-3) are modified. Clears the bit specified by the mask; i.e., bit 0, 1, 2, or 3. Bit 4 is a 0.

**CCC**

Clear all Condition Code Bits      CC                      000257

**Operation:**      N: 0  
                       Z: 0  
                       V: 0  
                       C: 0

**CLC**

Clear C                      CC                      000241

**Operation:**      N: unaffected  
                       Z: unaffected  
                       V: unaffected  
                       C: 0

**CLN**

Clear N                      CC                      000250

**Operation:**      N: 0  
                       Z: unaffected  
                       V: unaffected  
                       C: unaffected

**CLV**

Clear V                      CC                      000242

**Operation:** N: unaffected  
 Z: unaffected  
 V: 0  
 C: unaffected

**CLZ**

Clear Z CC 000244

**Operation:** N: unaffected  
 Z: 0  
 V: unaffected  
 C: unaffected

**CMP  
 CMPB**

Compare DO 02SSDD  
 12SSDD

**Operation:**  $(src) - (dst)$  [in detail  $(src) + \sim(dst) + 1$ ]

**Condition** N: set if result  $< 0$

**Codes:** 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: set if there is a borrow into the most  
 significant bit, i.e., if  $(src) + \sim(dst) + 1$  is less  
 than  $2^{16}$ .

**Description:** Compares the source and destination operands and sets the condition codes, which may then be used for arithmetic and logical conditional branches. Both operands are unaffected. The only action is to set the condition codes. The compare is customarily followed by a conditional branch instruction. Note that unlike the subtract instruction, the order of operation is  $(src) - (dst)$ , not  $(dst) - (src)$ .



**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 central processor status word (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 W7 on the KD11F module is inserted, a trap to 10 will occur unconditionally.

## NOTE

Execution of a HALT instruction causes the FALCON SBC-11/21 to stack PSW and PC, and set PC to restart address while setting PSW to 340<sub>8</sub>.





**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**            N: unaffected  
**Codes:**              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 maxi-

imum 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 of deferral less 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
<b>Operation:</b>	$SP \leftarrow PC + 2Xnn$ $nn = \text{number of parameters}$ $PC \leftarrow R5$ $R5 \leftarrow (SP) +$	
<b>Condition Codes:</b>	N: unaffected Z: unaffected V: unaffected C: unaffected	
<b>Description:</b>	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. This instruction is not available on FALCON SBC-11/21.	

**MFPD**  
**MFPI**

Move from Previous MS 1065SS  
Data Space 0065SS

Move from Previous  
Instruction Space

**Operation:** tmp ← (src)  
-(SP) ← tmp

**Condition Codes:** N: set if the source < 0  
Z: set if the source = 0  
V: cleared  
C: unaffected

**Description:** Pushes a word onto the current 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 MS 1067DD  
Processor  
Status Word

**Operation:** (dst) ← PS dst lower 8 bits

**Condition Codes:** N: set if PS bit 7 = 1  
Z: set if PS <7:0> = 0  
V: cleared  
C: unaffected

**Description:** The 8-bit contents of the PS are moved to the effective destination. If destination mode is 0, PS bit 7 is sign-extended through 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 only on the LSI-11/23.

**MFPT**

Move from Processor MS 000007

**Operation:** R0<7:0> ← processor model code  
R0<15:8> ← processor subcode

**Condition Codes:** N: unaffected  
Z: unaffected  
V: unaffected  
C: unaffected

**Description:** No source operands are used. The MFPT instruction returns in the low byte of R0 a processor model code. The high byte of R0 is loaded with a processor-specific subcode. The condition codes are not affected. The previous contents of R0 are lost.

Note: Standard for the LSI-/23 and FALCON SBC-11/21 only. On processors where this instruction is not implemented, a reserved instruction trap through vector 10<sub>6</sub> is taken.

**MOV  
MOVB**

Move DO 01SSDD  
11SSDD

**Operation:** (dst) ← (src)

**Condition Codes:** N: set if (src) < 0  
Z: set if (src) = 0  
V: cleared  
C: unaffected

**Description:** Moves the source operand to the destination location. The previous contents of the destination are lost. The source operand is not affected.

The MOV<sub>B</sub> 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 MOV<sub>B</sub> 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 perform 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:**       $tmp \leftarrow (SP) +$   
                          $(dst) \leftarrow tmp$

**Condition**      N: set if the source < 0  
**Codes:**        Z: set if the source = 0  
                         V: cleared  
                         C: unaffected

**Description:**    This instruction pops a word off the current stack determined by PS <15:14> and stores that word into an address in previous space PS <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 last 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**

N: set according to effective src operand 0-3

**Codes:**

Z: set according to effective src operand 0-3

V: set according to effective src operand 0-3

C: set according to effective src operand 0-3

**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 only on the LSI-11/23.

**NEG**  
**NEGB**

Negate

SO

0054DD

1054DD

**Operation:**

(dst) ← ~(dst) + 1

**Condition**

N: set if result &lt; 0

**Codes:**

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.

**NOP**

No Operation	CC	000240 000260
--------------	----	------------------

**Operation:** none

**Condition** N: unaffected

**Codes:** Z: unaffected

V: unaffected

C: unaffected

**Description:** No operation is performed.

**RESET**

Reset External Bus	MS	000005
--------------------	----	--------

**Operation:**

**Condition** N: unaffected

**Codes:** Z: unaffected

V: unaffected

C: unaffected

**Description:** Causes bus signal BINIT L to be asserted for 10  $\mu$ s and then unasserted for 90  $\mu$ s. 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)  $\leftarrow$  (dst) rotate left one place

**Condition** N: set if the high order bit of the result word is set (result > 0)

**Codes:**

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:**  $(dst) \leftarrow (dst)$  rotate right one place

**Condition**                      N: set if high-order bit of the result is set

**Codes:**                         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 destination.

**RTI**

Return from Interrupt            MS                      000002

**Operation:**                     $PC \leftarrow (SP) +$   
 $PS \leftarrow (SP) +$

**Condition**                      N: loaded from processor stack

**Codes:**                         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 nonre-entrant 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. Executes the same as the RT instruction with one exception. If 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:**  $(dst) \leftarrow (dst) - C$

**Condition** N: set if result < 0

**Codes:** 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
------------	----	--------

**Description:** 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
-------	----	--------

**Description:** 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	000270
-------	----	--------

**Description:** 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	000262
-------	----	--------

**Description:** 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
-------	----	--------

**Description:** 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:**       $R \leftarrow R - 1$  if this result does not = 0 then  $PC \leftarrow PC - (2 \times \text{offset})$

**Condition**      N: unaffected

**Codes:**      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:**       $(\text{dst}) \leftarrow (\text{dst}) - (\text{src})$

**Condition**      N: set if result < 0

**Codes:**      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:**                       $\text{tmp} \leftarrow (\text{dst})\langle 7:0 \rangle$   
     $(\text{dst})\langle 7:0 \rangle \leftarrow (\text{dst})\langle 15:8 \rangle$   
     $(\text{dst})\langle 15:8 \rangle \leftarrow \text{tmp}$

**Condition Codes:**                N: set if high order bit of low order byte (bit 7) of result is set  
    Z: set if low order byte of result = 0  
    V: cleared  
    C: cleared

**Description:**                      Exchanges high-order byte and low-order byte of the destination (which must be a word address).

## SXT

Sign Extend                      SO                      0067DD

**Operation:**                       $(\text{dst}) \leftarrow 0$  if N is clear  
     $(\text{dst}) \leftarrow -1$  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 and LSI-11/2 processors perform a DATIO cycle for the last bus cycle as a “don’t care” for hardware minimization.

**TRAP**

Trap	PC	104400 to 104777
<b>Operation:</b>	-(SP) ← PS -(SP) ← PC PC ← (34) PS ← (36)	
<b>Condition Codes:</b>	N: loaded from trap vector Z: loaded from trap vector V: loaded from trap vector C: loaded from trap vector	
<b>Description:</b>	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
<b>Operation:</b>	(dst) ← (dst)	
<b>Condition Codes:</b>	N: set if result < 0 Z: set if result = 0 V: cleared C: cleared	
<b>Description:</b>	Sets the condition codes N and Z according to the contents of the destination address.	

**WAIT**

Wait for Interrupt      MS      000001

**Operation:**

**Condition**      N: unaffected  
**Codes:**      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:**       $(dst) \leftarrow R \nabla (dst)$

**Condition**      N: set if the result  $< 0$   
**Codes:**      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.

**EXTENDED INSTRUCTION SET (EIS)****Summary of Extended Instruction Set**

ASH  
 ASHC  
 DIV  
 MUL

EIS instructions are standard on the LSI-11/23, PDP-11/23, and PDP-11/23-PLUS, and are available as an option on the:

LSI-11  
 LSI-11/2  
 PDP-11/03

**ASH**

Arithmetic Shift

DO

072RSS

**Operation:**

$R \leftarrow R$  shifted arithmetically NN places to the right or left where  $NN = (src) <5:0>$

**Condition**

N: set if result  $< 0$

**Codes:**

Z: set if result = 0

V: set if sign of register changed during shift

C: loaded from last bit shifted out of register. Cleared if  $NN=0$ .

**Description:**

Standard on LSI-11/23. Optional on LSI-11 and LSI-11/2. 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.

**ASHC**

Arithmetic Shift  
 Combined

DO

073RSS

**Operation:**

$tmp \leftarrow R, Rv1$

$tmp \leftarrow tmp$  shifted  
 NN bits

$R \leftarrow tmp <31:16>$

$RV1 \leftarrow tmp <15:0>$

The double word R,Rv1 is shifted NN places to the right or left, where  $NN = (src) <5:0>$



<b>Condition Codes:</b>	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 shift; loaded with low-order bit when right shift (loaded with the last bit shifted out of the 32-bit operand)
<b>Description:</b>	Optional on LSI-11 and LSI-11/2. The contents of the specified register R, and the register Rv1 are treated as a single 32-bit operand and are shifted by the number of bits specified by the count field (bits <5:0> of the source operand) and the registers are replaced by the result. First, bits <31:16> of the result are stored in register R. Then, bits <15:0> of the result are stored in register R1. The count ranges from -32 to +31. A negative count signifies a right shift. A positive count signifies a left shift. A zero count implies no shift, but condition codes are affected. Condition codes are always set on the 32-bit result.
<b>Note:</b>	1) The sign bit of the register R is replicated in shifts to the right. The least significant bit is filled with zero in shifts to the left. The C bit stores the last bit shifted out. 2) Integer overflow occurs on a left shift if any bit shifted into the sign position differs from the initial sign of the register. The register R is replicated in shifts to the right.
	<b>DIV</b>
Divide	DO 071RSS
<b>Operation:</b>	R,Rv1 ← R,Rv1/(src)
<b>Condition Codes:</b>	N: set if quotient < 0 (unspecified if V = 1) Z: set if quotient = 0 (unspecified if V = 1) V: set if source = 0 or if quotient cannot be represented as a 16-bit 2's complement number. R, Rv1 are unpredictable if V is set. C: set if divide by 0 attempted
<b>Description:</b>	Standard for LSI-11/23. Optional for LSI-11/2 and LSI-11. 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 in Rv1 is of the same sign as the dividend. R must be even.

**MUL**

Multiply

DO

070RSS

**Operation:** $R, Rv1 \leftarrow R \times (src)$ **Condition**N: set if product  $< 0$ **Codes:**

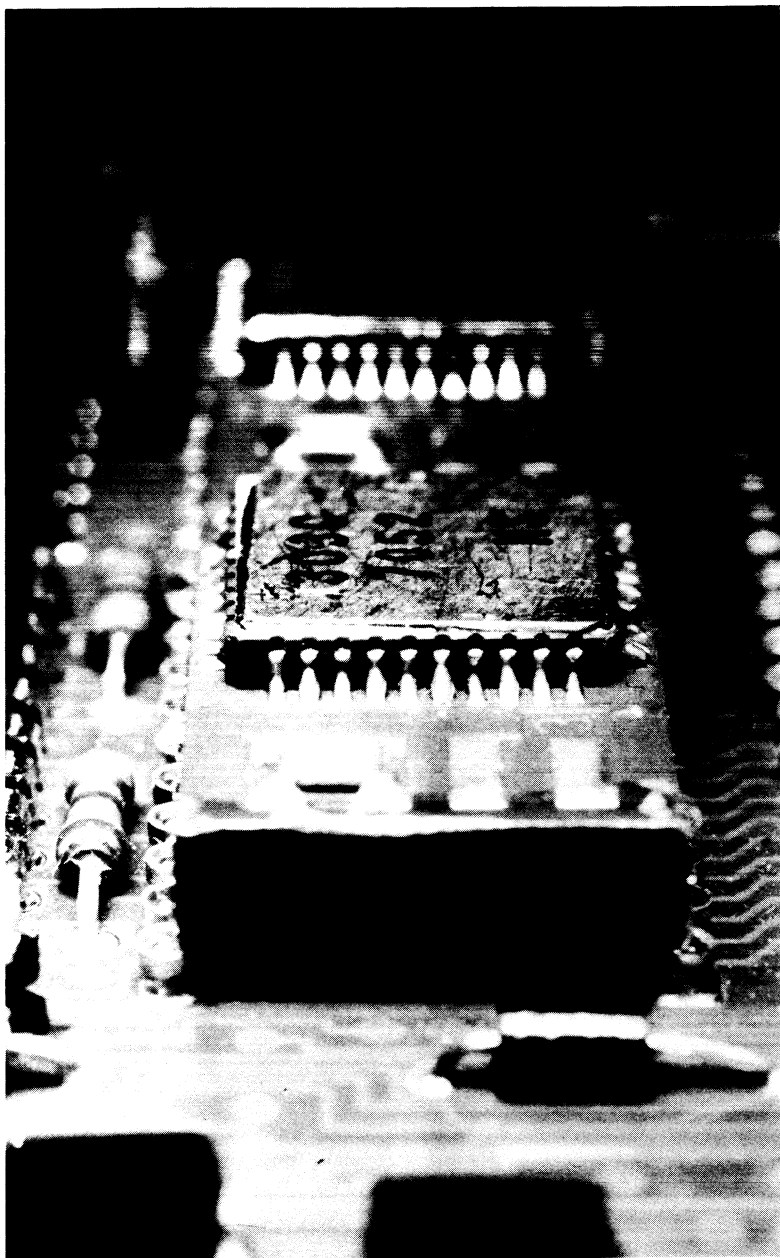
Z: set if product = 0

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.)





**KEF11-AA Floating Point Option**

# FLOATING POINT INSTRUCTION SET FP-11 (LSI-11/23, PDP-11/23, AND PDP-11/23-PLUS)

## INTRODUCTION

The LSI-11 microcomputer family has two sets of floating point instructions- the Floating Point Option Instruction Set (FP-11) and the Floating Point Instruction Set (FIS). For a discussion on the Floating Point Instruction Set available with the LSI-11, LSI-11/2, and PDP-11/03, please refer to Chapter six.

The FP-11 Instruction Set, available for the LSI-11/23, PDP-11/23, and PDP-11/23-PLUS as the microcode option KEF11-AA or in the FPF11 option, supports both single- and double-precision floating point arithmetic. The FP-11 instruction set does not include FIS instructions. The floating point option functions as an integral part of the central processor. It uses similar address modes and uses the memory management facilities provided by the memory management option. FP-11 instructions can reference the floating point accumulators, the central processor's general registers, or any location in memory. Figure 5-1 illustrates the conceptual structure of the floating point option.

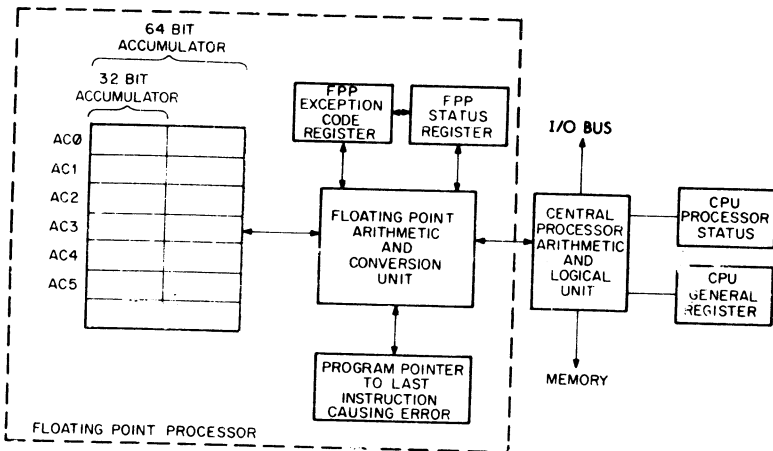


Figure 5-1 Conceptual Structure of the Floating Point Option

### NOTE

The KEF11-AA option requires the MMU to be installed on the LSI-11/23 since it utilizes registers present on the MMU chip in executing instructions. The PPF11 option contains all the necessary floating point accumulators, and therefore does not require the MMU unit to be installed.

### 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 FP-11 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 are 32 bits long. In double-precision, or double mode, the data are 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 fractions 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  $\frac{1}{2}$ . Therefore, the floating point option reserves a biased expo-

ment 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 floating point option 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  $-0$  occurs, it did not result from a previous floating point arithmetic instruction.

### Floating Point Data

Floating point data are stored in words of memory as illustrated in Figures 5-2 and 5-3.

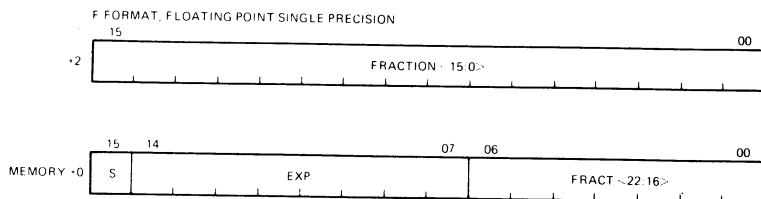


Figure 5-2 Single-Precision Format

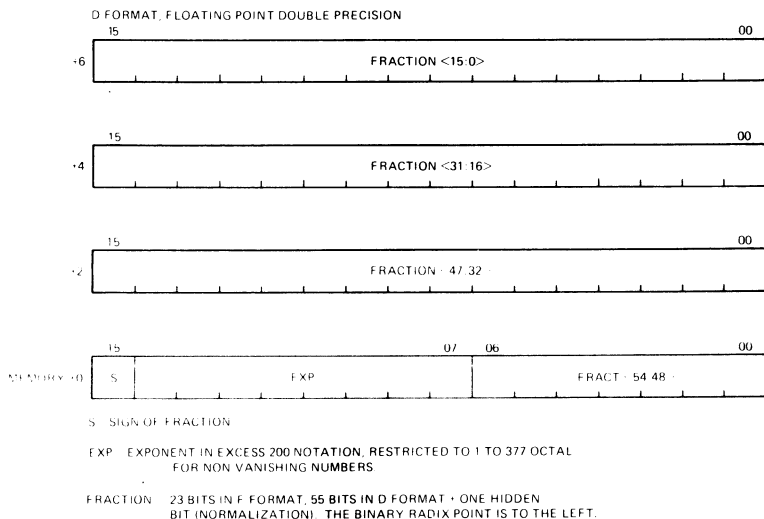


Figure 5-3 Double-Precision Format

The floating point option 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. The 2's complement format is illustrated in Figure 5-4.

### 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. The floating point status register is illustrated in Figure 5-5.

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.
- Normal/Maintenance: A special maintenance mode is available on the FP11-C and FP11-E.



## Chapter 5 — Floating Point Instruction Set

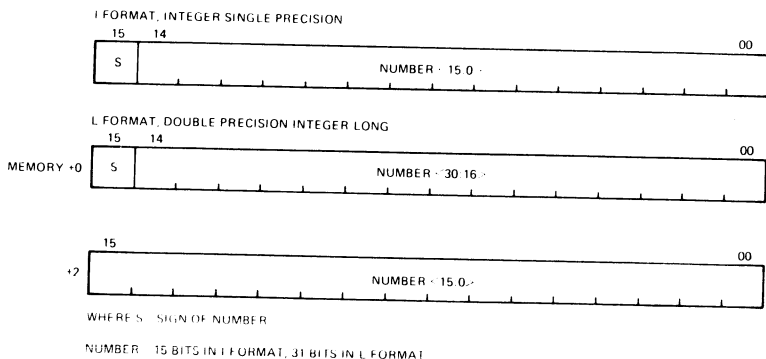


Figure 5-4 2's Complement Format

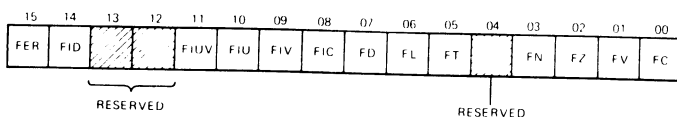


Figure 5-5 Floating Point Status Register

The FPS register contains an error flag and four condition codes (five bits): carry, overflow, zero, and negative, which are equivalent to the CPU condition codes.

The floating point processor recognizes seven floating point exceptions:

- Detection of the presence of the undefined variable in memory
- Floating overflow
- Floating underflow
- Failure of floating to integer conversion
- Maintenance trap (FP11-C, FP11-E only)
- Attempt to divide by zero
- Illegal floating opcode

For the first five of these exceptions, bits in the FPS register are available to enable or disable interrupts individually. An interrupt on the occurrence of either of the last two exceptions can be disabled only by setting a bit which disables interrupts on all seven of the exceptions as a group.

Of the fourteen bits described above, five, the error flag and condition codes, are set by the FPP as part of the output of a floating point instruction. Any of the mode and interrupt control bits (except the FP11-C and FP11-E, FMM bit) may be set by the user; the LDFS instruction is available for this purpose. These 14 bits are stored in the FPS register as follows:

#### **FPS Register Bits**

**Bit: 15      Name:** Floating Error (FER)

**Function:** The FER bit is set by a floating point instruction 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 FPP never resets the FER bit. Once the FER bit is set by the FPP, 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.

**Bit: 14      Name:** Interrupt Disable (FID)

**Function:** 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 FPP is always accompanied 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.

**Bit: 13**

**Function:** Reserved for future DIGITAL use.

**Bit: 12**

**Function:** Reserved for future DIGITAL use.

**Bit: 11      Name:** Interrupt on Undefined Variable (FIUV)

**Function:** 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 exe-

cution on the floating point option except on NEG, ABS, and TST1, for which it occurs after execution. When FIUV is reset,  $-0$  can be loaded and used in any floating point option 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 FPP will not store a result of  $-0$  without a simultaneous interrupt.

**Bit: 10 Name:** Interrupt on Underflow (FIU)

**Function:** 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.

**Bit: 9 Name:** Interrupt on Overflow (FIV)

**Function:** 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 FPP returns exact 0.

Special cases of overflow are discussed in the detailed descriptions of the MOD and LDEXP instructions.

**Bit: 8 Name:** Interrupt on Integer Conversion Error (FIC)

**Function:** 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 6).

**Bit: 7 Name:** Floating Double-Precision Mode (FD)

**Function:** 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.

**Bit: 6 Name:** Floating Long Integer Mode (FL)

**Function:** 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).

**Bit: 5      Name:** Floating Chop Mode (FT)

**Function:** When the FT bit is set, the result of any arithmetic operation is chopped (or truncated). When reset, the result is rounded.

**Bit: 4      Name:** Floating Maintenance Mode (FMM)

**Function:** FP11-C and FP11-E only. When set, the FPP is in maintenance mode. The FMM bit can be set only in kernel mode.

**Bit: 3      Name:** Floating Negative (FN)

**Function:** FN is set if the result of the last floating point operation was negative, otherwise it is reset.

**Bit: 2      Name:** Floating Zero (FZ)

**Function:** FZ is set if the result of the last floating point operation was 0, otherwise it is reset.

**Bit: 1      Name:** Floating Overflow (FV)

**Function:** FV is set if the last floating point operation resulted in an exponent overflow, otherwise it is reset.

**Bit: 0      Name:** Floating Carry (FC)

**Function:** 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). The six possible errors are coded in the 4-bit floating exception code (FEC) register as follows:

2	Floating op code error
4	Floating divide by zero
6	Floating or double to integer conversion error
8	Floating overflow
10	Floating underflow
12	Floating undefined variable
14	Maintenance trap (FP11-C, and FP11-E only)

The address of the instruction producing the exception is stored in the FEA (Floating Exception Address) register.

The FEC and FEA registers are updated when one of the following occurs:

- Divide by zero
- Illegal op code
- Any of the other five exceptions with the corresponding interrupt enabled

If one of the five 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 are not 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 OPTION INSTRUCTION ADDRESSING

Floating point option 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 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 floating point accumulators (AC0-AC5) as source or destination. Specifying floating point option 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 floating point 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 1 least significant bit (LSB). A rounded result has an error bound of  $\frac{1}{2}$  LSB. These error bounds are realized by the LSI-11/23 for all instructions. The FP11-A, the FP11-E, and the FP11-F have an error bound greater than  $\frac{1}{2}$  LSB for ADD and SUB. For the addition of operands of opposite sign or for the subtraction of operands of the same sign in rounded double-precision, the error bound is  $\frac{3}{4}$  LSB (FP11-C and FP11-E), or  $\frac{33}{64}$  LSB (FP11-A and FP11-F) which is slightly larger than the  $\frac{1}{2}$  LSB error bound for all other rounded operations.

The error bound for the FP11-C differs from the FP11-A, since the FP11-C and FP11-E carry three guard bits while the FP11-A and FP11-F carry seven guard bits.

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 stored correctly 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 floating point option 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 FIV (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 enabled, 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 occurrence of underflow or overflow.

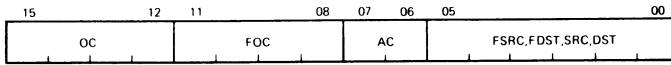
## FLOATING POINT INSTRUCTIONS

Each instruction that manipulates a floating point number can operate on either single- or double-precision numbers, depending on the state of FD mode bit. Similarly, there is a mode bit FL that determines whether 32-bit integers or 16-bit integers are used in conversion between integer and floating point representation. FSRC and FDST use floating point addressing modes; SRC and DST use CPU addressing modes. Figure 5-6 illustrates single- and double-floating point operand addressing.

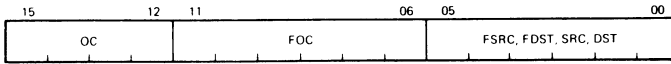
In the descriptions of the floating point instructions, the operations of the KEF11-AA, PPF11, FP11-A, FP11-E, FP11-F, and FP11-C are identical, except where explicitly stated otherwise. Table 5-1 describes the floating point conventions used in the PDP-11 instruction set.

## Chapter 5 — Floating Point Instruction Set

### DOUBLE OPERAND ADDRESSING



### SINGLE OPERAND ADDRESSING



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

Figure 5-6 Single- and Double-Operand Addressing

Table 5-1 Floating Point Conventions

Mnemonic	Description
OC	Opcode = 17
FOC	Floating Opcode
AC	Contents of accumulator, as specified by AC field of instruction
FSRC	Address of floating point source operand.
FDST	Address of floating point destination operand
f	Fraction
XL	Largest fraction that can be represented: $1 - 2^{**}(-24)$ , FD=0, single-precision $1 - 2^{**}(-56)$ , FD=1; double-precision
XLL	Smallest number that is not identically zero $= 2^{**}(-128)$
XUL	Largest number that can be represented = $2^{**}(127)*XL$
JL	Largest integer that can be represented: $2^{**}(15)-1$ if FL=0, $2^{**}(31)-1$ if FL=1
ABS[(x)]	Absolute value of contents of memory location X



EXP[(x)]	Biased exponent of contents of memory location X
<	Less than
≤	Less than or equal to
>	Greater than
≥	Greater than or equal to
≠	Not equal to
LSB	Least significant bit

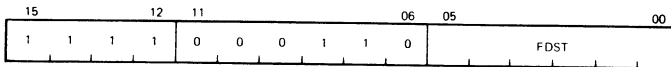
### FP-11 Floating Point Instructions

#### ABSF

#### ABSD

Make Absolute Floating/Double

1706 FDST



**Format:** ABSF FDST

**Operation:** If  $(fdst) < 0$ ,  $(fdst) \leftarrow -(fdst)$ .  
 If  $EXP[(fdst)] = 0$ ,  $(fdst) \leftarrow \text{exact } 0$ .  
 For all other cases,  $(fdst) \leftarrow (fdst)$ .

**Condition** FC  $\leftarrow 0$

**Codes:** FV  $\leftarrow 0$

FZ  $\leftarrow 1$  if  $(fdst) = 0$ , else FZ  $\leftarrow 0$

FN  $\leftarrow 0$

**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.

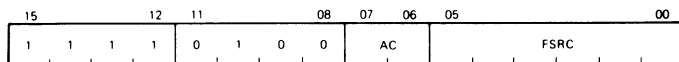
**Accuracy:** These instructions are exact.

**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  $\leftarrow 1$ ).

**ADDF**  
**ADDD**

Add Floating/Double

172(AC)FSRC

**Format:** ADDF FSRC,AC**Operation:** Let  $SUM = AC + (fsrc)$ . If underflow occurs and FIU is not enabled,  $AC \leftarrow \text{exact } 0$ .If overflow occurs and FIV is not enabled,  $AC \leftarrow \text{exact } 0$ .For all other cases,  $AC \leftarrow SUM$ .**Condition**  $FC \leftarrow 0$ **Codes:**  $FV \leftarrow 1$  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:** 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:

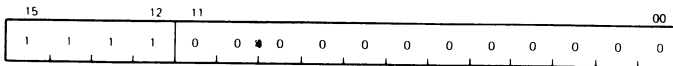
1. 1 LSB in truncated mode with either single- or double-precision.
2.  $\frac{1}{2}$  LSB in rounding mode with either single (all FP-11s, KEF11-AA and FPF11) or double-precision (for LSI-11/23 floating point options);  $\frac{3}{4}$  LSB (FP11-C and FP11-E) or  $33/64$  LSB (FP11-A, FP11-F and FPF11) in rounding mode with 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.

**CFCC**

Copy Floating Condition Codes

170000



**Format:** CFCC

**Operation:** C ← FC  
 V ← FV  
 Z ← FZ  
 N ← FN

**Description:** Copy the floating point condition codes into the CPU's condition codes.

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  $\frac{1}{2}$  LSB in rounding mode.

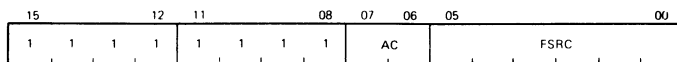
**Special Comment:**

The undefined variable  $-0$  can occur only in conjunction with overflow and underflow. It will be stored in AC only if the corresponding interrupt is enabled.

**LDCDF****LDCFD**

Load and Convert from Double to Floating  
and from Floating to Double

177(AC+4)FSRC

**Format:**

LDCDF      FSRC,AC

**Operation:**

If  $EXP[(fsrc)] = 0$ ,  $AC \leftarrow \text{exact } 0$ .

If  $FD = 1$ ,  $FT = 0$ ,  $FIV = 0$  and rounding causes overflow,  $AC \leftarrow \text{exact } 0$ .

In all other cases,  $AC \leftarrow Cxy[(fsrc)]$ , 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:**  $FC \leftarrow 0$   
 $FV \leftarrow 1$  if conversion produces overflow, 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 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.  $AC \leftarrow$  overflowed result. This result must be  $+0$  or  $-0$ .

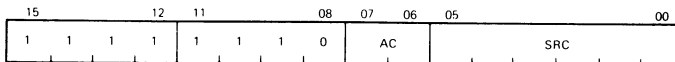
Underflow cannot occur.

**Accuracy:** LDCFD is an exact instruction. Except for overflow, described above, LDCDF incurs an error bounded by 1 LSB in chopping mode and by  $\frac{1}{2}$  LSB in rounding mode.

**LDCIF LDCLF**  
**LDCID LDCLD**

Load and Convert Integer or Long Integer  
to Floating or Double Precision

177(AC)SRC



**Format:** LDCIF SRC,AC

**Operation:**  $AC \leftarrow Cjx[(src)]$ , where  $Cjx$  specifies conversion from integer mode  $j$  to floating mode  $y$ .

$$j = I \text{ if } FL = 0, j = L \text{ if } FL = 1$$

$$x = F \text{ if } FD = 0, x = D \text{ if } FD = 1$$

**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.

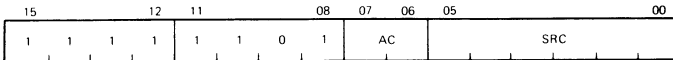
In the case of LDCLF, the fractional part of the floating point representation is chopped or rounded to 24 bits according to the state of FT (1 = chop, 0 = round).

**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.

**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,  $EXP[AC] \leftarrow [(src) + 200_8] < 7:0 >$  on the FP11-A,

FP11-E,  
FP11-F, KEF11-AA, and FPF11.

$EXP[AC] \leftarrow (src) \langle 6:0 \rangle$  on the FP11-C.

If  $(src) > 177_8$  and FIV is disabled,  $AC \leftarrow \text{exact } 0$ .

If  $(src) < -177_8$  and FIU is enabled,

$EXP[AC] \leftarrow [(src) + 200_8] \langle 7:0 \rangle$  on the FP11-A,  
FP11-E,

FP11-F, KEF11-AA, and FPF11.

$EXP[AC] \leftarrow (src) \langle 6:0 \rangle$  on the FP11-C.

If  $(src) < -177_8$  and FIU is disabled,  $AC \leftarrow \text{exact } 0$ .

**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  $ABS[(src)] \leq 177_8$ .

If  $(src) > 177_8$ , the result is treated as overflow. If

$(src) < -177_8$ , the result is treated as underflow.

Note that the KEF11-AA does not treat these abnormal conditions as the FP11-C and FP11-B do, but it does treat them as the FP11-A, FP11-E, FP11-F, and FPF11 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  $(src) \neq -200$ , 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

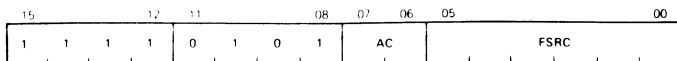
of arithmetic instructions is triggered by a non-vanishing value of EXP.

For all other cases, LDEXP implements exactly the transformation of a floating point number ( $2^{**}K$ ) \* f into  $(2^{**}(src)) * f$  where  $\frac{1}{2} \leq ABS(f) < 1$ .

**LDF**  
**LDD**

Load Floating/Double

172 (AC+4)FSRC



**Format:** LDF FSRC,AC

**Operation:** AC ← (fsrc)

**Condition** FC ← 0

**Codes:** 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.

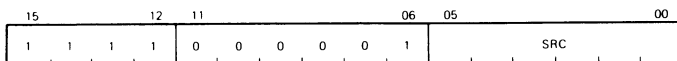
**Accuracy:** These instructions are exact.

**Special Comment:** These instructions permit use of -0 in a subsequent floating point instruction if FIUV is not enabled and (fsrc) = -0.

**LDFPS**

Load FPP Program Status

1701 SRC



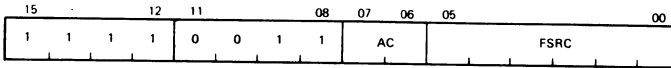


**Format:** LDFPS SRC  
**Operation:** FPS ← (src)  
**Description:** Load KEF11-AA's status register from (src).  
**Special Comment:** Bits 13, 12, and 4 should not be used for the user's own purposes, since these bits are not recoverable by the STFPS instruction. Bit 4 may be set in kernel mode if the KEF11-AA implements maintenance mode.

**MODF  
MODD**

Multiply and Separate Integer  
and Fraction Floating/Double

171(AC+4)FSRC



**Format:** MODF FSRC,AC  
**Description and Operation:** This instruction generates the product of its two floating point operands, separates the product into integer and fractional parts, and then stores one or both parts as floating point numbers.  
 Let PROD = AC \* (fsrc) so that in  
 Floating point: ABS [PROD] = (2\*\*K) \* f  
 • where  
 $\frac{1}{2} \leq f < 1$  and  
 EXP[PROD] = (200 + K) octal  
 Fixed point binary: PROD = N + g with  
 N = INT[PROD] = the integer part of PROD  
 and  
 g = PROD - INT[PROD] = the fractional part of PROD with  $0 \leq g < 1$ .  
 Both N and f 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 ACv1 and g is returned to AC, where v means OR.

Five special cases occur, as indicated in the following formal description with  $L = 24$  for floating mode and  $L = 56$  for double mode.

1. If PROD overflows and FIV is enabled, ACv1  $\leftarrow$  N, chopped to L bits, AC  $\leftarrow$  exact 0.

Note that EXP[N] is too small by  $400_8$ , and that  $-0$  can get stored in ACv1.

If FIV is not enabled, ACv1  $\leftarrow$  exact 0, AC  $\leftarrow$  exact 0, and  $-0$  will never be stored.

2. If  $2^{**L} \leq \text{ABS}[\text{PROD}]$  and no overflow, ACv1  $\leftarrow$  N, chopped to L bits, AC  $\leftarrow$  exact 0.

The sign and EXP of N are correct, but low-order bit information is lost.

3. If  $1 \leq \text{ABS}[\text{PROD}] < 2^{**L}$ , ACv1  $\leftarrow$  N, AC  $\leftarrow$  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  $\pm$  unity for the fractional part. For  $L = 24$ , the error in g is bounded by 1 LSB in chopping mode and by  $\frac{1}{2}$  LSB in rounding mode. For  $L = 56$ , the error in g increases from the above limits as ABS[N] increases above  $2^{**L}$  because only 59 bits (64 bits for KEF11-AA) of PROD are generated.

If  $2^{**p} \leq \text{ABS}[N] < 2^{**(p+1)}$ , with  $p > 2$  (7 for KEF11-AA) the low-order  $p-2$  ( $p-7$  for KEF11-AA) bits of g may be in error.

4. If  $\text{ABS}[\text{PROD}] < 1$  and no underflow, ACv1  $\leftarrow$  exact 0 and AC  $\leftarrow$  g.

There is no error in the integer part. The error in the fractional part is bounded by 1 LSB in chopping mode and  $\frac{1}{2}$  LSB in rounding mode. Rounding may cause a return of  $\pm$  unity for the fractional part.

5. If PROD underflows and FIU is enabled, ACv1  $\leftarrow$  exact 0 and AC  $\leftarrow$  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  $\leftarrow$  exact 0 and AC  $\leftarrow$  exact 0.

For this case the error in the fractional part is less than  $2^{*-}(-128)$ .

**Condition**

FC  $\leftarrow$  0

**Codes:**

FV  $\leftarrow$  1 if PROD overflows, else FV  $\leftarrow$  0

FZ  $\leftarrow$  1 if AC = 0, else FZ  $\leftarrow$  0

FN  $\leftarrow$  1 if AC < 0, else FN  $\leftarrow$  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  $\leftarrow$  0;

X  $\leftarrow$  number

to

be converted;

ABS[X] < 1;

While X  $\neq$  0 do

Begin PROD  $\leftarrow$  X \* 10;

I  $\leftarrow$  I + 1;

D (I)  $\leftarrow$  INT(PROD);

X  $\leftarrow$  PROD - INT(PROD);

End;

This algorithm is exact. It is case 3 in the description because the number of nonvanishing bits in the fractional part of PROD never exceeds L, and hence neither chopping nor rounding can introduce error.

2. To reduce the argument of a trigonometric function.

$ARG * 2/\pi = N + g$ . The low two bits of  $N$  identify the quadrant, and  $g$  is the argument reduced to the first quadrant. The accuracy of  $N+g$  is limited to  $L$  bits because of the factor  $2/\pi$ . The accuracy of the reduced argument thus depends on the size of  $N$ .

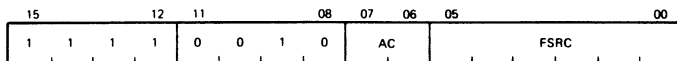
3. To evaluate the exponential function  $e^{**x}$ , obtain  $x * (\log e \text{ base } 2) = N + g$ , then  $e^{**x} = (2^{**N}) * (e^{**g * 1n 2})$ .

The reduced argument is  $g * 1n 2 < 1$  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  $(\log e \text{ base } 2)$ . The accuracy of the reduced argument thus depends on the size of  $N$ .

**MULF  
MULD**

Multiply Floating/Double

171(AC)FSRC



**Format:** MULF FSRC,AC

**Operation:** Let  $PROD = AC * (fsrc)$ .

If underflow occurs and FIU is not enabled,  $AC \leftarrow$  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) ← exact 0. For all other cases, PROD is generated to 48 (32 for KEF11-AA) bits for floating mode and 59 (64 for KEF11-AA) bits for double mode. The product is rounded or chopped according to the value of the FT bit, and is stored in AC except for:

1. Overflow with interrupt disabled.
2. Underflow with interrupt disabled.

For these exceptional cases, an exact 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.

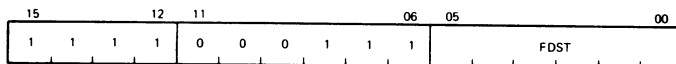
**Accuracy:** Errors due to overflow and underflow are described above. If neither occurs, the error incurred is bounded by 1 LSB in chopping mode and  $\frac{1}{2}$  LSB in rounding mode.

**Special Comment:** The undefined variable  $-0$  can occur only in conjunction with overflow or underflow. It will be stored in AC only if the corresponding interrupt is enabled.

**NEGF**  
**NEGD**

Negate Floating/Double

1707 FDST



**Format:** NEGF  $-(fdst)$

**Operation:**  $(fdst) \leftarrow -(fdst)$  if EXP  $[(fdst)] \neq 0$ , else  $(fdst) \leftarrow$  exact 0.

**Condition**  $FC \leftarrow 0$   
**Codes:**  $FV \leftarrow 0$   
 $FZ \leftarrow 1$  if (fdst) = 0, else  $FZ \leftarrow 0$   
 $FN \leftarrow 1$  if (fdst) < 0, else  $FN \leftarrow 0$

**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.

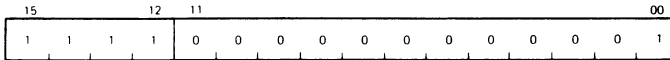
**Accuracy:** These instructions are exact.

**Special Comment:** If a  $-0$  is present in memory and the FIUV bit is enabled, then the KEF11-AA stores an exact zero in memory. If a negative number is present, then the PPF11 stores the actual negative result in memory. The condition codes reflect an exact 0 ( $FZ \leftarrow 1$ ).

**SETF**

Set Floating Mode

170001

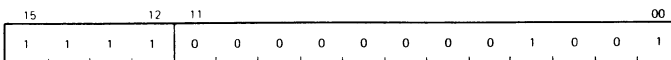


**Format:** SETF  
**Operation:**  $FD \leftarrow 0$   
**Description:** Set the floating point option in single-precision mode.

**SETD**

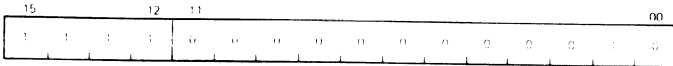
Set Floating Double Mode

170011



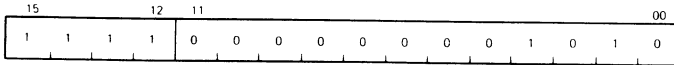
**Format:** SETD  
**Operation:** FD ← 1  
**Description:** Set the floating point option in double-precision mode.

**SETI**  
Set Integer Mode 177002



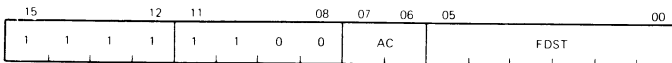
**Format:** SETI  
**Operation:** FL ← 0  
**Description:** Set the floating point option for short integer data.

**SETL**  
Set Long Integer Mode 177012



**Format:** SETL  
**Operation:** FL ← 1  
**Description:** Set the floating point option for long integer data.

**STCFD**  
**STCDF**  
Store and Convert from Floating to Double  
and from Double to Floating 176(AC)FDST



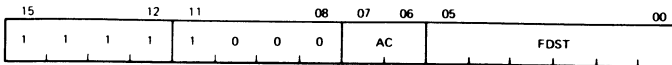
<b>Format:</b>	STCFD AC,FDST
<b>Operation:</b>	<p>If <math>AC = 0</math>, <math>(fdst) \leftarrow \text{exact } 0</math>.</p> <p>If <math>FD = 1</math>, <math>FT = 0</math>, <math>FIV = 0</math> and rounding causes overflow, <math>(fdst) \leftarrow \text{exact } 0</math>.</p> <p>In all other cases, <math>(fdst) \leftarrow C_{xy}[AC]</math>, where <math>C_{xy}</math> specifies conversion from floating mode <math>x</math> to floating mode <math>y</math>.</p> <p style="padding-left: 40px;"><math>x = F, y = D</math> if <math>FD = 0</math> (single) STCFD  <math>x = D, y = F</math> if <math>FD = 1</math> (double) STCDF</p>
<b>Condition Codes:</b>	<p><math>FC \leftarrow 0</math></p> <p><math>FV \leftarrow 1</math> if conversion produces overflow, else <math>FV \leftarrow 0</math></p> <p><math>FZ \leftarrow 1</math> if <math>AC = 0</math>, else <math>FZ \leftarrow 0</math></p> <p><math>FN \leftarrow 1</math> if <math>AC &lt; 0</math>, else <math>FN \leftarrow 0</math></p>
<b>Description:</b>	<p>If the current mode is single-precision, the accumulator is stored left-justified in FDST and the lower half is cleared.</p> <p>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.</p>
<b>Interrupts:</b>	<p>Trap on <math>-0</math> will not occur even if FIUV is enabled because FSRC is an accumulator.</p> <p>Underflow cannot occur.</p> <p>Overflow cannot occur for STCFD.</p> <p>A trap occurs if FIV is enabled, and if rounding with STCDF causes overflow. <math>(fdst) \leftarrow \text{overflowed result}</math>. This must be <math>+0</math> or <math>-0</math>.</p>
<b>Accuracy:</b>	STCFD is an exact instruction. Except for overflow, described above, STCDF incurs an error bounded by 1 LSB in chopping mode and by $\frac{1}{2}$ LSB in rounding mode.



**STF  
STD**

Store Floating/Double

174(AC)FDST

**Format:** STF AC,FDST**Operation:** (fdst) ← AC**Condition** FC ← FC**Codes:** 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.

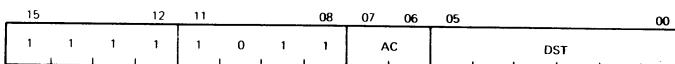
Overflow and underflow cannot occur.

**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-AA. One occurs when underflow or overflow is present and the corresponding interrupt is enabled. A second occurs when an LDF, LDD, LDCDF, or LDCFD instruction is executed and the FIUV bit is disabled.

**STCFI STCDI  
STCFL STCDL**Store and Convert from Floating or Double  
to Integer or Long Integer

175(AC+4)DST

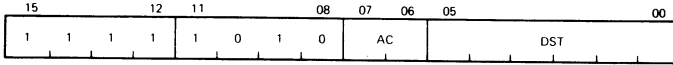


<b>Format:</b>	STCFI AC,DST
<b>Operation:</b>	$(dst) \leftarrow Cxj[AC]$ if $-JL-1 < Cxj[AC] < JL+1$ , else $(dst) \leftarrow 0$ , where $Cjx$ specifies conversion from floating mode $x$ to integer mode $j$ . $j = I$ if $FL = 0$ , $j = L$ if $FL = 1$ $x = F$ if $FD = 0$ , $x = D$ if $FD = 1$ $JL$ is the largest integer $2^{15}-1$ for $FL = 0$ $2^{32}-1$ for $FL = 1$
<b>Condition Codes:</b>	$C, FC \leftarrow 0$ if $-JL-1 < Cxj[AC] < JL+1$ , else $C, FC \leftarrow 1$ $V, FV \leftarrow 0$ $Z, FZ \leftarrow 1$ if $(dst) = 0$ , else $Z, FZ \leftarrow 0$ $N, FN \leftarrow 1$ if $(dst) < 0$ , else $N, FN \leftarrow 0$
<b>Description:</b>	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.
<b>Interrupts:</b>	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.
<b>Special Comment:</b>	These instructions store the integer part of the floating point operand, which may not be the integer most closely approximating the operand. They are exact if the integer part is within the range implied by $FL$ .

**STEXP**

Store Exponent

175(AC)DST

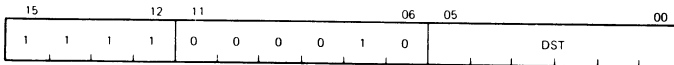


- Format:** STEXP AC,DST
- Operation:**  $(dst) \leftarrow EXP[AC] - 200_8$
- Condition** C, FC  $\leftarrow 0$
- Codes:** V, FV  $\leftarrow 0$   
 Z, FZ  $\leftarrow 1$  if  $(dst) = 0$ , else Z, FZ  $\leftarrow 0$   
 N, FN  $\leftarrow 1$  if  $(dst) < 0$ , else N, FN  $\leftarrow 0$
- Description:** Convert AC's exponent from excess  $200_8$  notation to 2's complement and store the result in dst.
- Interrupts:** This instruction will not trap on  $-0$ .  
 Overflow and underflow cannot occur.
- Accuracy:** This instruction is always exact.

**STFPS**

Store KEF11AA's Program Status

1702 DST

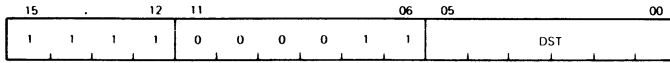


- Format:** STFPS DST
- Operation:**  $(dst) \leftarrow FPS$
- Description:** Store floating point status register in dst.
- Special Comment:** Bits 13, 12, and 4 (if maintenance mode is not implemented) are loaded with 0. All other bits are the corresponding bits in the FPS.

**STST**

Store KEF11-AA's Status

1703 DST



**Format:** STST DST

**Operation:** (dst) ← FEC  
(dst + 2) ← FEA

**Description:** Store the FEC and FEA in dst and dst+2.

**NOTE**

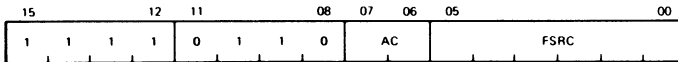
1. If the destination mode specifies a general register or immediate addressing, only the FEC is saved.
2. The information in these registers is current only if the most recently executed floating point instruction caused a floating point exception.

**SUBF**

**SUBD**

Subtract Floating/Double

173(AC)FSRC



**Format:** SUBF FSRC,AC

**Operation:** Let DIFF = AC - (fsrc).  
 If underflow occurs and FIU is not enabled, AC ← exact 0.  
 If overflow occurs and FIV is not enabled, AC ← exact 0.  
 For all cases, AC ← DIFF.

<b>Condition Codes:</b>	$FC \leftarrow 0$ $FV \leftarrow 1$ if overflow occurs, else $FV \leftarrow 0$ $FZ \leftarrow 1$ if $AC = 0$ , else $FZ \leftarrow 0$ $FN \leftarrow 1$ if $AC < 0$ , else $FN \leftarrow 0$
<b>Description:</b>	<p>Subtract the contents of <i>fsrc</i> from the contents of <i>AC</i>. The subtraction is carried out in single- or double-precision and is rounded or chopped according to the values of the <i>FD</i> and <i>FT</i> bits in the <i>FPS</i> register. The result is stored in <i>AC</i> except for:</p> <ol style="list-style-type: none"> <li>1. Overflow with interrupt disabled.</li> <li>2. Underflow with interrupt disabled.</li> </ol> <p>For these exceptional cases, an exact 0 is stored in <i>AC</i>.</p>
<b>Interrupts:</b>	<p>If <i>FIUV</i> is enabled, trap on <math>-0</math> in <i>fsrc</i> occurs before execution.</p> <p>If overflow or underflow occurs and if the corresponding interrupt is enabled, the trap occurs with the faulty result in <i>AC</i>. The fractional parts are correctly stored. The exponent part is too small by <math>400_8</math> for overflow. It is too large by <math>400_8</math> for underflow, except for the special case of 0, which is correct.</p>
<b>Accuracy:</b>	<p>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:</p> <ol style="list-style-type: none"> <li>1. 1 LSB in truncated mode with either single- or double-precision</li> <li>2. <math>\frac{1}{2}</math> LSB in rounding mode with either single- (all <i>FP-11s</i>, <i>KEF11-AA</i> and <i>FPF11</i>) or double-precision (<i>KEF11-AA</i> only); <math>\frac{3}{4}</math> LSB (<i>FP11-C</i> and <i>FP11-E</i>) and <math>\frac{33}{64}</math> LSB (<i>FP11-A</i>, <i>FP11-F</i> and <i>FPF11</i>) in rounding mode with double-precision.</li> </ol>

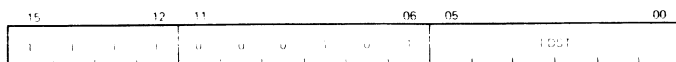
**Special Comment:** The undefined variable  $-0$  can occur only in conjunction with overflow or underflow. It will be stored in AC only if the corresponding interrupt is enabled.

**TSTF**

**TSTD**

Test Floating/Double

1705 FDST



**Format:** TSTF FDST

**Operation:** (fdst)

**Condition** FC  $\leftarrow$  0

**Codes:** FV  $\leftarrow$  0

FZ  $\leftarrow$  1 if (fdst) = 0, else FZ  $\leftarrow$  0

FN  $\leftarrow$  1 if (fdst) < 0, else FN  $\leftarrow$  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.







# FLOATING POINT INSTRUCTION SET FIS (LSI-11, LSI-11/2, AND PDP-11/03)

## INTRODUCTION

The Floating Point Instruction Set (FIS) option consists of four instructions: Floating Add (FADD), Floating Subtract (FSUB), Floating Multiply (FMUL), and Floating Divide (FDIV). These instructions operate on single-precision floating formats, and are available on the LSI-11, LSI-11/2, and PDP-11/03 only. The KEV11 is the EIS/FIS option for the LSI-11, LSI-11/2, and PDP-11/03.

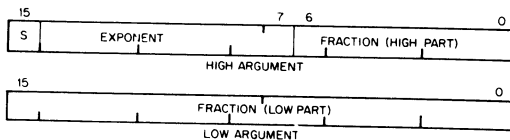
## KEV11 OPTION

### FIS Instruction Set

The following floating point instruction opcodes do not conflict with any other instructions and are not compatible with the FP-11 Instruction Set.

Mnemonic	Instruction	Op code
FADD	Floating Add	07500R
FSUB	Floating Subtract	07501R
FMUL	Floating Multiply	07502R
FDIV	Floating Divide	07503R

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 FP-11 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  $\frac{1}{2}$  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:

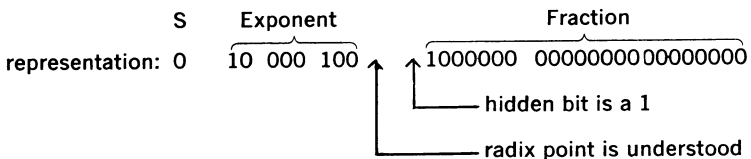
Actual Value	Representation	
Decimal	Octal	Binary
+127	377	11 111 111
+1	201	10 000 001
0	200	10 000 000
-1	177	01 111 111
-128	000	00 000 000

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**

$$+(12)_{10} = +(1100)_2$$

$$= +(2^4)_{10} \times (.11)_2 \qquad [16 \times (\frac{1}{2} + \frac{1}{4}) = 12]$$



**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, i.e., R is incremented by 4.

### Condition Codes

Condition codes are set or cleared as shown in the instruction descriptions, in the next part of this section. If a trap occurs as a function of a floating point instruction, the condition codes are reinterpreted as follows:

V = 1, if an error occurs  
 N = 1, if underflow 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 because of 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.

**FIS Instructions**

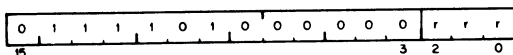
Assembler format is: OPR R

(R) denotes contents of memory location whose address is in R.

**FADD**

Floating Add

07500R



**Operation:**

$$[(R+4), (R+6)] \leftarrow [(R+4), (R+6)] + [(R), (R+2)]$$

**Condition**

N: set if result < 0; cleared otherwise

**Codes:**

Z: set if result = 0; cleared otherwise

V: cleared

C: cleared

**Description:**

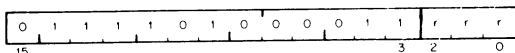
Adds the A argument to the B argument and stores the result in the A argument position on the stack. General register R is used as the stack pointer for the operation.

$$A \leftarrow A + B$$

**FDIV**

Floating Divide

07503R



**Operation:**  $[(R+4), (R+6)] \leftarrow [(R+4), (R+6)] / [(R), (R+2)]$

**Condition** N: set if result < 0; cleared otherwise

**Codes:** 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.

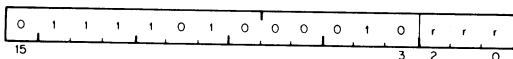
$A \leftarrow A/B$

**Note:** The LSI-11 processors push one word onto the stack during execution of the FMUL and FDIV instructions and pop the word from the stack when completed. Thus, the SP (R6) must point to a read/write memory location; otherwise, a bus error (time-out) will occur.

## FMUL

Floating Multiply

07502R



**Operation:**  $[(R+4), (R+6)] \leftarrow [(R+4), (R+6)] \times [(R), (R+2)]$

**Condition** N: set if result < 0; cleared otherwise

**Codes:** 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.

$A \leftarrow A \times B$

(refer to note in FDIV)

**FSUB**

Floating Subtract

07501R

**Operation:**  $[(R+4), (R+6)] \leftarrow [(R+4), (R+6)] - [(R), (R+2)]$

**Condition** N: set if result < 0; cleared otherwise

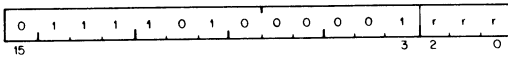
**Codes:** Z: set if result = 0; cleared otherwise

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.

$A \leftarrow A - B$









## CHAPTER 7

# OCTAL DEBUGGING TECHNIQUE (MICROCODE ODT)

### CONSOLE ODT

The console emulator Octal Debugging Technique (ODT) is a portion of the processor microcode that is very useful for debugging and running programs. Console emulator ODT allows the processor to respond to commands and information entered via the computer console or from a local or remote terminal. Communication between the user and processor is generated via a stream of ASCII characters interpreted by the processor as console commands. These commands are a subset of ODT-11. (Please refer to Table 7-1 below).

Terminal addresses utilized by console emulator ODT are 177560<sub>8</sub> through 177566<sub>8</sub> (777560<sub>8</sub> through 777566<sub>8</sub> for 18-bit systems and 1777560<sub>8</sub> through 1777566<sub>8</sub> for 22-bit systems). These addresses are generated in microcode and cannot be altered.

To determine the hardware requirements necessary to run console emulator ODT, please refer to the Hardware Requirement section at the end of this chapter. If you are currently using DIGITAL standard DLV11 serial line interfaces configured at the console address, these hardware requirements have been met and there is no need to access this information.

The console ODT implemented on the LSI-11/23, PDP-11/23, and PDP-11/23-PLUS is identical. For chapter readability and clarity, only the LSI-11/23 is referenced. The reader may assume that all references to the LSI-11/23, also apply to the PDP-11/23 and PDP-11/23-PLUS.

### CONSOLE ODT COMMANDS

The console ODT terminal command set, listed in Table 7-1, is described in the following paragraphs. These commands are a subset of ODT-11 and use the same command characters. Console ODT has ten internal states, which are listed in Table 7-2. For each state, only specific characters are recognized as valid inputs; other inputs invoke a "?" response.

**Table 7-1 Console ODT Commands**

<b>Command</b>	<b>Symbol</b>	<b>Use</b>
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 \$ or R command.
Go	G	Starts program execution.
Proceed	P	Resumes execution of a program.
Binary Dump	Control-Shift-S	Manufacturing use only.
	H	Reserved for DIGITAL use.

**Table 7-2 Console ODT States and Valid Input Characters**

<b>State</b>	<b>Example of Terminal Output</b>	<b>Valid Input</b>	<b>Comment</b>
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>	
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 by ODT (e.g., <CR>) 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).

### 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 time-out 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 time-out during refresh cannot take place. This differs from the LSI-11, which enters console ODT upon such an occurrence. If a bus time-out 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 commands, 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 OPERATION

The processor's microcode operates the serial line interface in half-duplex mode. Programmed 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.

### LSI-11/23 ODT Commands

#### **/(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 are 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 yet 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 64 KB segment and issues <LF>, console ODT opens the first location of that same segment. If the user wishes to cross the 64 KB boundary, he must re-enter the address for the desired 64 KB segment (i.e., console ODT is modulo 64 KB). 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:    @\$0/054321<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 single-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:    @200G<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 initialization 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, are cleared to 0.
4. The LSI-11 Bus is initialized by the processor's 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 re-enters 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:     @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 service 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 it is accidentally invoked. It is intended to display a portion of memory more efficiently than 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 <15:8> and the second byte specifies starting address <7:0>. Bus address bits <17:16> are always forced to be 0; the dump command is restricted to the first 64 KB 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 (248KB to 256 KB) 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 an MMU present, 18-bit addresses must be used to access memory greater than 64 KB. In 22-bit systems, console ODT allows access to physical memory 000000 through 757776 and the I/O page (17760000 through 17777776) only. The I/O page is accessed by entering addresses 760000 to 777776.

### 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 to bus addresses 777700 through 777707 if referenced in console ODT (i.e., time-out occurs).

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

Example:     @777572/000001 <SPACE>

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

### Stack Pointer Selection

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.

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> was 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 it 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 Time-Out**

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's unintentionally destroying a program by pressing the wrong key.

### **LSI-11 and LSI-11/2 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 (time-out) occurs during memory refresh while in the console ODT mode, a ?<CR><LF>@ will be typed.

### **/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: @ 001000/012525

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.

### **<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 six digits entered were placed in location 1000.

### <LF> Line Feed (ASCII 012)

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. <LF> will modify the PS if new data have 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.

### ↑ 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.

**@ 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 are typed, and close it; however, the last GPR or memory location content will be used as a pointer.

**← 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 have been typed, and close the GPR or PS; then a <CR><LF>@ will be issued.

### **\$ 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.

### **\$\$ (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 = 8-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.

### **G (ASCII 107) Go**

The G command is used 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 to act as fill characters in case the 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. 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 BHALT 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.

#### **P (ASCII 120) Proceed**

The P 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 BHALT 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 compatibility. If the semicolon character is received during any character sequence, the console ignores it.

#### **M (ASCII 115) Maintenance**

The M command is used for maintenance purposes and prints the contents of an internal CPU register. This data reflect 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.

<b>Last Octal Digital Value</b>	<b>Function</b>
0 or 4	HALT instruction or BHALT L.
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.
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.

### **RO (ASCII 177) RUBOUT**

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>  
              @000100/123456

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 reflect the RUBOUT. Note that if RUBOUT is issued repeatedly, only numerical characters are erased. 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.

### **L (ASCII 114) Bootstrap Loader**

The L 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 56K bytes (157776), and the address is decremented by 2 until the highest read/write memory location is found. In small systems (e.g., 8K bytes memory), a discernible pause of about 1 second will occur before type motion is observed. 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 be 352<sub>8</sub>. The Absolute Loader program tape, for example, has several inches of frames all punched with 351<sub>8</sub>. The first byte following the 351<sub>8</sub> bytes contains the low byte of the starting address minus 1. (For Absolute Loader, this byte is 075<sub>8</sub>.) All bytes which follow are data bytes. Loading continues until address XXX752 has been loaded. The data at that location are 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 PS 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 time-out 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.

#### 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>@.

#### HARDWARE REQUIREMENTS

The minimum hardware requirements for a serial line interface to permit 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<sub>8</sub> 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. The receiver status register is illustrated in Figure 7-1.

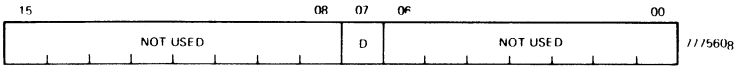


Figure 7-1 Receiver Status Register

**Bit: 7      Name: Done flag**

**Function:** 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.

**Bit: 6:0, 15:8      Name: Unused**

**Function:** 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<sub>8</sub> for character input to console ODT. This register needs to respond to DATI bus cycles only, 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. The receiver buffer register is illustrated in Figure 7-2.

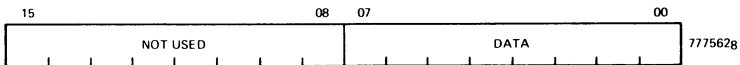


Figure 7-2 Receiver Buffer Register

**Bit: 7:0      Name: ASCII character**

**Function:** 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.

**Bit: 15:8 Name:** Unused

**Function:** 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<sub>8</sub> 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). The transmitter control and status register is illustrated in Figure 7-3.

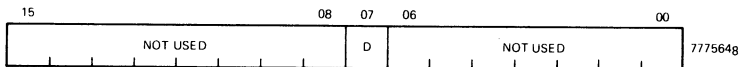


Figure 7-3 Transmitter Control and Status Register

**Bit: 7 Name:** Done flag

**Function:** 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.

**Bit: 6:0**

**15:8 Name:** Unused

**Function:** 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<sub>8</sub> for character output from console ODT. This register needs to respond to DATO bus cycles only, 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. The transmitter buffer register is illustrated in Figure 7-4.

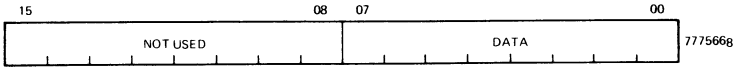


Figure 7-4 Transmitter Buffer Register

**Bit: 7:0 Name:** ASCII character

**Function:** 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.

**Bit: 15:8 Name:** Unused

**Function:** 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.







## CHAPTER 8

# PROGRAMMING TECHNIQUES

The LSI-11 and PDP-11 microcomputers offer you a great deal of programming flexibility and power. With the combination of the instruction set, addressing modes, and programming techniques, new software can be developed and old programs utilized effectively. This chapter provides programming techniques examples which illustrate the unique capabilities of PDP-11 processors. The techniques specifically discussed are: position-independent coding (PIC), stacks, subroutines, interrupts, re-entrancy, 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 PDP-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 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 PDP-11 addressing modes. The remainder of this explanation assumes that you are familiar with the addressing modes described in Chapter 3.

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 nonrelocatable 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 may be position-*dependent*. Therefore, 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
MOV      $OTSV,R4       ;SET R1=R2=SP
                          ;GET OTS IMPURE
MOV      (SP)+,R2       ;AREA POINTER
                          ;GET AST ENTRY
BNE     10$             ;POINT ADDRESS
                          ;IF NONE SPECIFIED,
                          ;SPECIFY NO POWER
CLR      -(SP)           ;RECOVERY AST SERVICE
BR       20$            ;
```

```

10$:      MOV     R2,F.PF(R4)      ;
          MOV     #BA,-(SP)      ;SET AST ENTRY POINT
          ;PUSH AST SERVICE
          ;ADDRESS
20$:      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$:      MOV     R2,F.PF(R4)    ;
          MOV     PC,-(SP)      ;SET AST ENTRY POINT
          ADD     #BA-.,(SP)     ;PUSH CURRENT LOCATION
          ;COMPUTE ACTUAL
          ;LOCATION
          ;OF AST
20$:      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 LSI-11 and PDP-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 a PDP-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

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 argu-

ments 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:

Address	Octal Code	Assembler 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	016702	MOV TEMP2,R2	;restore R2
076416	000004	*	
076420	000207	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.

Address	Octal Code	Assembler 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 MOV<sub>B</sub> (MOVE BYTE) or INC (INCREMENT) instruction. This example is illustrated in Figure 8-3.

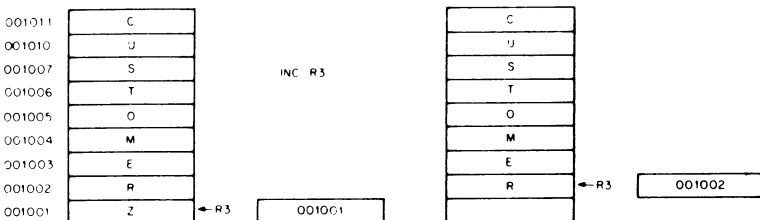


Figure 8-3 Byte Stack Used as a Character Buffer



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 a 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.

Figure 8-4 illustrates the before and after conditions when executing the subroutine instructions JSR R5,1064.

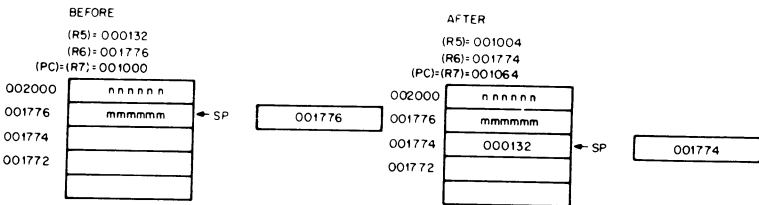


Figure 8-4 JSR Instruction

Because the PDP-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, effected 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.

### **Caution When Clearing Device Interrupt Enable Bits**

Clearing device Interrupt Enable bits while the device is still active can lead to a bus time-out error when the processor attempts to receive the interrupt vector from that device. Consider the example:

```
PSW = 0
CLR @ #177564
```

As a result, the DLV-11 Serial Line Unit Interrupt Enable bit is being cleared. Now, assume that the transmitter is still active and sending characters, and further assume that the Done bit in the status register becomes set shortly after the CLR instruction is fetched, but before the Interrupt Enable bit can be cleared. The device will now post an interrupt request because the Done bit has been set and the Interrupt Enable bit is still set. The CLR instruction will complete execution and the processor will recognize the interrupt request since there was not enough time for the device to disable the interrupt request. The proc-

essor will then attempt to obtain a vector from the interrupting device. However, a bus time-out error will occur because the device now has had enough time to remove the interrupt request and will not respond. The processor treats this time-out as a fatal condition and halts by entering Micro-ODT. If multiple interrupt requests were pending at this time, a time-out would not occur since the next device would respond with an interrupt vector.

One method of avoiding this problem is to disable interrupts immediately before the Interrupt Enable bit is cleared. For example:

```

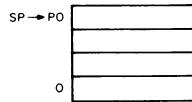
MTPS #200
CLR @#177564
MTPS #0
    
```

In this situation, enough time has been allowed for the interrupt request to be removed by the device. This feature was included to detect faulty interrupt operation; specifically when an interrupting device does not properly respond within the required time period.

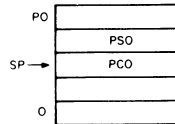
### Nesting

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. Nested interrupt service routines and subroutines are illustrated in Figure 8-5.

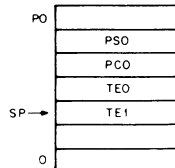
1. Process 0 is running; SP is pointing to location P0.



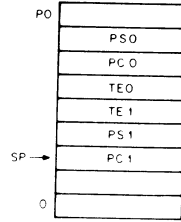
2. Interrupt stops process 0 with PC = PC0, and status = PS0; starts process 1.



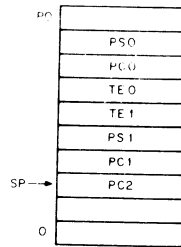
3. Process 1 uses stack for temporary storage (TE0, TE1).



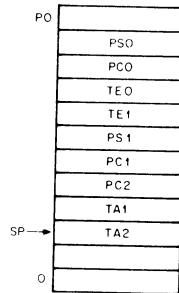
4. Process 1 interrupted with PC = PC1 and status = PS1; process 2 is started.



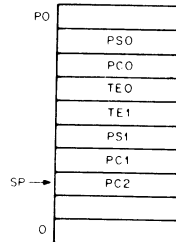
5. Process 2 is running and does a JSR R7,A to subroutine A with PC = PC2.



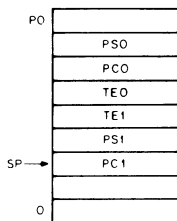
6. Subroutine A is running and uses stack for temporary storage.



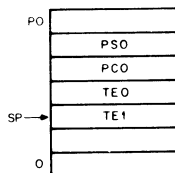
7. Subroutine A releases the temporary storage holding TA1 and TA2.



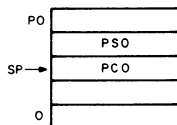
8. Subroutine A returns control to process 2 with an RTS R7; PC is reset to PC2.



9. Process 2 completes with an RTI instruction (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.

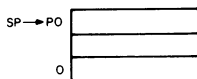


Figure 8-5 Nested Interrupt Service Routines and Subroutines

Note that the area of interrupt service programming is intimately involved with the concept of CPU and device priority levels.

### RE-ENTRANCY

Other advantages of the PDP-11 stack organization are evidenced in programming systems that are engaged in concurrent handling of

several tasks. Multitask 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 multiprogramming systems that manage an intricate mixture of executive and multiuser programming situations, as in RSX-11. In all cases, 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 **re-entrancy**. Re-entrant program routines differ from ordinary subroutines in that it is not necessary for re-entrant 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. Figure 8-6 illustrates this situation.

Approach

Programs 1, 2, and 3 can share Subroutine A.

Conventional Approach

A separate copy of Subroutine A must be provided for each program.



Figure 8-6 Re-entrant Routines

### Re-entrant Code

Re-entrant 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 re-entrant. 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

Using re-entrant code, control of a routine can be shared as illustrated in Figure 8-7.

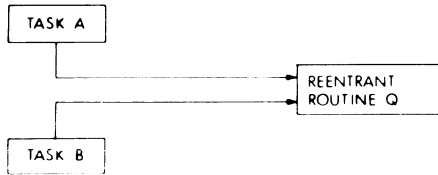


Figure 8-7 Sharing Control of a Routine

- Task A requests processing by Re-entrant Routine Q.
- Task A temporarily relinquishes control of Re-entrant Routine Q before it completes processing.
- Task B starts processing the same copy of Re-entrant Routine Q.
- Task B completes processing by Re-entrant Routine Q.
- Task A regains use of Re-entrant Routine Q and resumes where it stopped.

### Writing Re-entrant 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, re-entrant 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, thereby causing a new area to be used as the stack when the second task is entered.



The following should be observed when writing re-entrant 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 program, it should be by indexed addresses, which can be set by the calling task in order to handle any possible recursion.

### Use of Re-entrant Code

Re-entrant 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 PDP-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. Figure 8-8 illustrates a coroutine example.

For example:

Routine A	Stack	Routine B	Comments
.		.	LOC is pushed
.		.	onto the stack
.		.	to prepare for
.MOV #LOC, -(SP)	LOC ← SP		the coroutine
			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.

Figure 8-8 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.
- 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.

Figure 8-9 illustrates the structure of subroutines and coroutines.

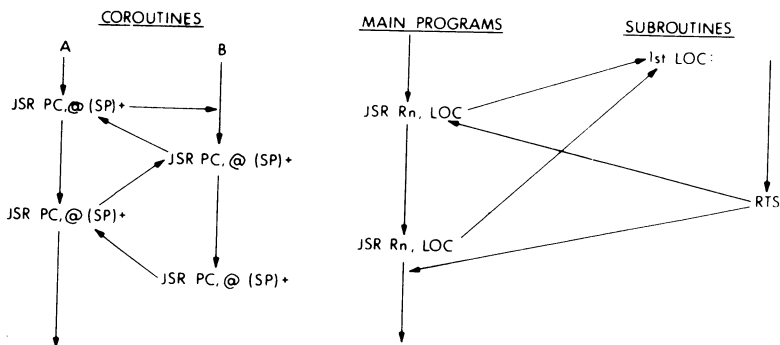


Figure 8-9 Coroutines vs. Subroutines

**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 make debugging easier.

**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 used effectively here with one routine being the assembly-pass-one routine and the other extracting one item at a time from the current input line. This situation is illustrated in Figure 8-10.

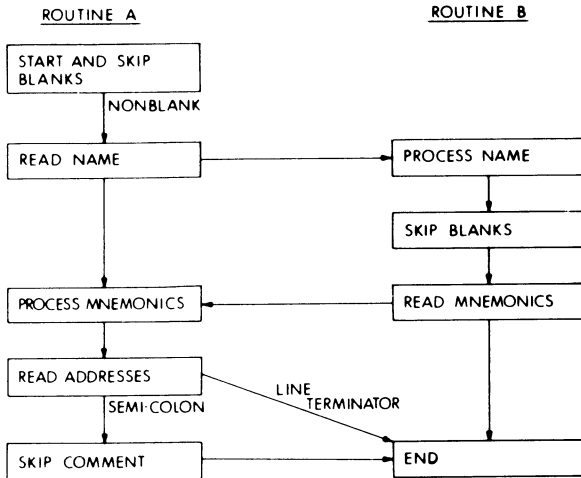


Figure 8-10 Coroutine Path

Coroutines can be utilized in I/O processing. The example shows coroutines used in double-buffered I/O using IOX. This example is illustrated in Figure 8-11. The flow of events might be described as:

```

Write 01
Read I1           concurrently
Process I2

then

Write 02
Read I2           concurrently
Process I1
    
```

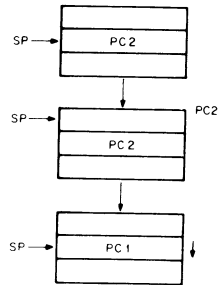
Routine #1 is operating; it then executes:

```

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.

Figure 8-11 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 subroutine 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 is illustrated in Figure 8-12.

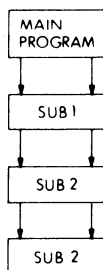


Figure 8-12 Recursive Routine Flow

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 DDCF 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 programs shorter 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:

$$XM = 1 + F(X)$$

and:

$$F(X) = G(X)$$

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. When a macro call is encountered within a 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 is 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 nonexistent 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	Power-fail
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 needs to 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, time-out)
	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**

Nearly 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.

Nearly 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
    
```

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.

$$\begin{array}{r} 221 \\ -1000 \end{array}$$

Negative, so go to next lower power, count for  $10^3=0$ .

$$\begin{array}{r} 221 \\ -100 \end{array}$$

$$\begin{array}{r} 121 \\ -100 \end{array} \quad \text{count for } 10^2=1.$$

$$\begin{array}{r} 21 \\ -100 \end{array} \quad \text{count} = 2$$

Negative, so reduce power.

Count for  $10^2=2$

$$\begin{array}{r} 21 \\ -10 \end{array}$$

$$11 \quad \text{count for } 10^1=1.$$

$$\begin{array}{r} 11 \\ -10 \end{array}$$

$$\begin{array}{r} 1 \\ -10 \end{array} \quad \text{count}=2$$

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<sub>8</sub> (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;

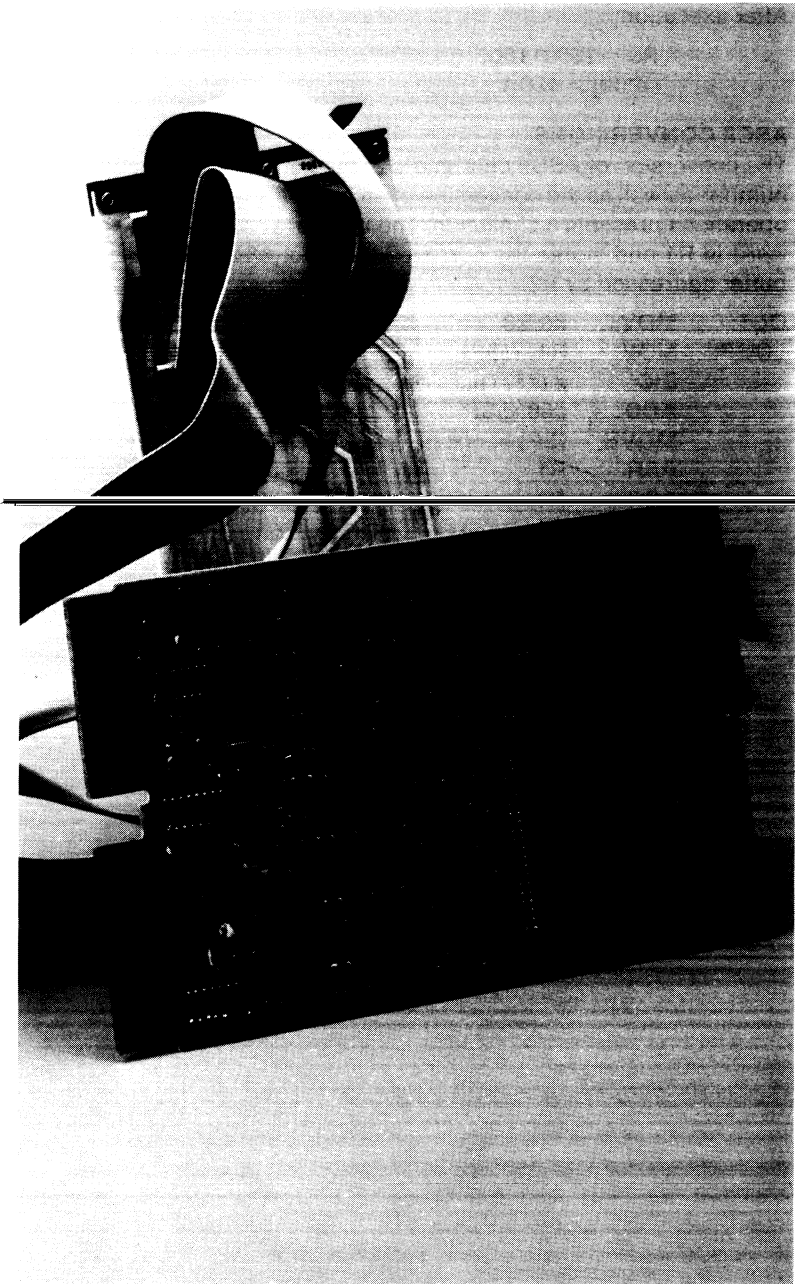
R0 = 100011000  
 ( $7 \cdot 50_8 = 430_8$ ).

### 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     #60,@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     #60,R1         ;CONVERT TO ASCII
       MOVB   R1,-(R2)       ;STORE IN BUFFER
       RTS    PC              ;DONE,RETURN
  
```





## CHAPTER 9

# LSI-11 BUS

The LSI-11 Bus is the low-end member of DIGITAL's bus family. All DIGITAL microcomputers use the LSI-11 Bus. However, in order to use the 22-bit addressing capabilities of the LSI-11/23, PDP-11/23, and the PDP-11/23-PLUS, the extended LSI-11 Bus is required.

The LSI-11 Bus consists of 42 bidirectional and 2 unidirectional signal lines. These form the lines along which the processor, memory, and I/O devices communicate with each other.

Addresses, data, and control information are sent along these signal lines, some of which contain time-multiplexed information. The lines are divided as follows:

- Sixteen multiplexed data/address lines — BDAL <15:00>
- Two multiplexed address/parity lines — BDAL <17:16>
- Four extended address lines — BDAL <21:18>
- Six data transfer control lines — BBS7, BDIN, BDOUT, BRPLY, BSYNC, BWTBT
- Six system control lines — BHALT, BREF, BEVNT, BINIT, BDCOK, BPOK
- Ten interrupt control and direct memory access control lines — BIAKO, BIAKI, BIRQ4, BIRQ5, BIRQ6, BIRQ7, BDMGO, BDMR, BSACK, BDMGI

In addition, a number of power, ground, and spare lines have been defined for the bus. For a detailed description of these lines, please refer to Table 9-1.

The discussion in this chapter applies to the general 22-bit physical address capability. In cases where modules utilize 16- or 18-bit physical address space, this discussion applies to the lines that are utilized by those modules.

Most LSI-11 Bus signals are bidirectional and use terminations for a negated (high) signal level. Devices connect to these lines via high-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., which becomes bus master at any given time. A typical example of this relationship is the processor, as master, fetching an instruction from memory, which is always a slave. Another example is a disk, as master, transferring data to memory as slave. Communication on the LSI-11 Bus is interlocked so that for certain control signals issued by the master device, there must be a response from the slave in order to complete the transfer. It is the master/slave signal protocol that makes the LSI-11 Bus asynchronous. The asynchronous operation precludes the need for synchronizing with, and waiting for, clock pulses.

Since bus cycle completion by the bus master requires response from the slave device, each bus master must include a time-out error circuit that will abort the bus cycle if the slave device does not respond to the bus transaction within 10 microseconds.

The actual time before a time-out error occurs must be longer than the reply time of the slowest peripheral or memory device on the bus. The signal assignments are shown in Table 9-1.

Table 9-1 Signal Assignments

**DATA AND ADDRESS**

<b>Nomenclature</b>	<b>Pin Assignment</b>
BDAL0	AU2
BDAL1	AV2
BDAL2	BE2
BDAL3	BF2
BDAL4	BH2
BDAL5	BJ2
BDAL6	BK2
BDAL7	BL2
BDAL8	BM2
BDAL9	BN2
BDAL10	BP2
BDAL11	BR2
BDAL12	BS2
BDAL13	BT2
BDAL14	BU2
BDAL15	BV2
BDAL16	AC1
BDAL17	AD1
BDAL18	BC1
BDAL19	BD1
BDAL20	BE1
BDAL21	BF1

**CONTROL**

<b>Nomenclature</b>	<b>Pin Assignment</b>
	<b>Data Control</b>
BDOUT	AE2
BRPLY	AF2
BDIN	AH2
BSYNC	AJ2
BWTBT	AK2
BBS7	AP2
	<b>Interrupt Control</b>
BIRQ7	BP1
BIRQ6	AB1
BIRQ5	AA1

BIRQ4	AL2
BIAK0	AN2
BIAK1	AM2
	<b>DMA Control</b>
BDMR	AN1
BSACK	BN1
BDMG0	AS2
BDMG1	AR2
	<b>System Control</b>
BHALT	AP1
BREF	AR1
BEVNT	BR1
BINIT	AT2
BDCOK	BA1
BPOK	BB1

**POWER AND GROUND**

<b>Nomenclature</b>	<b>Pin Assignment</b>
+5B	AS1
+12B(or battery)	
+12B	BS1
+5B	AV1
+5	AA2
+5	BA2
+5	BV1
+12	AD2
+12	BD2
-12	AB2
-12	BB2
GND	AC2
BSYNCL	AJ1
GND	AM1
GND	AT1
GND	BC2
GND	BJ1
GND	BM1
GND	BT1

**SPARES**

<b>Nomenclature</b>	<b>Pin Assignment</b>
SSpare1	AE1
SSpare3	AH1
SSpare8	BH1
SSpare2	AF1
MSpareA	AK1
MSpareB	AL1
MSpareB	BK1
MSpareB	BL1
PSpare1	AU1
ASpare2	BU1

**DATA TRANSFER BUS CYCLES**

Data transfer bus cycles are listed and defined in Table 9-2.

**Table 9-2 Data Transfer Operations**

<b>Bus Cycle Mnemonic</b>	<b>Description</b>	<b>Function (with Respect to the Bus Master)</b>
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 bus signals listed in Table 9-3 are used in the data transfer operations described in Table 9-2.

**Table 9-3 Bus Signals For Data Transfers**

<b>Mnemonic</b>	<b>Description</b>	<b>Function</b>
BDAL<21:00> L	22 Data/address lines	BDAL<15:00> L are used for word and byte transfers. BDAL<17:16> L are used for extended addressing, memory parity error (16), and memory parity error enable (17) functions. BDAL<21:18> L are used for extended addressing beyond 256 KB.
BSYNC L	Bus Cycle Control	Strobe signals.
BDIN L	Data input indicator	
BDOUT L	Data output indicator	
BRPLY L	Slave's acknowledge 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 re-

sponds by latching the address bits and holding this condition for the duration of the bus cycle until BSYNC L becomes negated. During the data transfer portion, the actual data transfer occurs.

**Device Addressing** — The device addressing portion of a data transfer bus cycle comprises an address setup and deskew time and an address hold and deskew time. During the address setup and deskew time, the bus master does the following:

- Asserts BDAL <21:00> L with the desired slave device address bits
- Asserts BBS7 L if a device in the I/O page is being addressed
- Asserts BWTBT L if the cycle is a DATO(B) bus cycle

During this time the address, BBS7 L, and BWTBT L signals are asserted at the slave bus receiver for at least 75 ns before BSYNC goes active. Devices in the I/O page ignore the nine high-order address bits BDAL <21:13> and instead decode BBS7 L along with the thirteen low-order address bits. An active BWTBT L signal indicates that a DATO(B) operation follows, while an inactive BWTBT L indicates a DATI or DATIO(B) operation.

The address hold and deskew time begins after BSYNC L is asserted.

The slave device uses the active BSYNC L bus receiver output to clock BDAL address bits, BBS7 L, and BWTBT L into its internal logic. BDAL <21:00> L, BBS7 L, and BWTBT L will remain active for 25 ns (minimum) after BSYNC L bus receiver goes active. BSYNC L remains active for the duration of the bus cycle.

Memory and peripheral devices are addressed similarly except for the way the slave device responds to BBS7 L. Addressed peripheral devices must not decode address bits on BDAL <21:13> L. Addressed peripheral devices may respond to a bus cycle when BBS7 L is asserted (low) during the addressing portion of the cycle. When asserted, BBS7 L indicates that the device address resides in the I/O page (the upper 4K address space). Memory devices generally do not respond to addresses in the I/O page; however, some system applications may permit memory to reside in the I/O page for use as DMA buffers, read-only memory bootstraps or diagnostics, etc.

**DATI** — The DATI bus cycle, illustrated in Figure 9-1, is a read operation. During DATI, data are input to the bus master. Data consist of 16-bit word transfers over the bus. During the data transfer portion of the DATI bus cycle, the bus master asserts BDIN L 100 ns minimum after BSYNC L is asserted. The slave device responds to BDIN L active as follows:

- Asserts BRPLY L after receiving BDIN L and 125 ns (maximum) before BDAL bus driver data bits are valid

- Asserts BDAL<21:00> L with the addressed data and error information

When the bus master receives BRPLY L, it does the following:

- Waits at least 200 ns deskew time and then accepts input data at BDAL<17:00> L bus receivers. BDAL<17:16> L are used for transmitting parity errors to the master
- Negates BDIN L 200 ns (minimum) to 2 microseconds (maximum) after BRPLY L goes active

The slave device responds to BDIN L negation by negating BRPLY L and removing read data from BDAL bus drivers. BRPLY L must be negated 100 ns (maximum) prior to removal of read data. The bus master responds to the negated BRPLY L by negating BSYNC L

Conditions for the next BSYNC L assertion are as follows:

- BSYNC L must remain negated for 200 ns (minimum)
- BSYNC L must not become asserted within 300 ns of previous BRPLY L negation

Figure 9-2 illustrates DAT1 bus cycle timing.

#### NOTE

Continuous assertion of BSYNC L retains control of the bus by the bus master, and the previously addressed slave device remains selected. This is done for DAT1O(B) bus cycles where DATO or DATOB follows a DAT1 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.



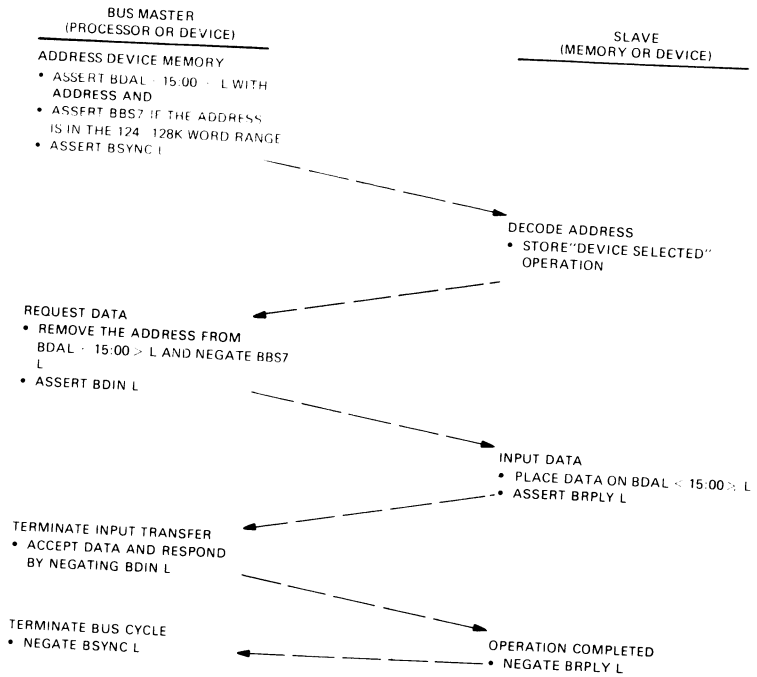
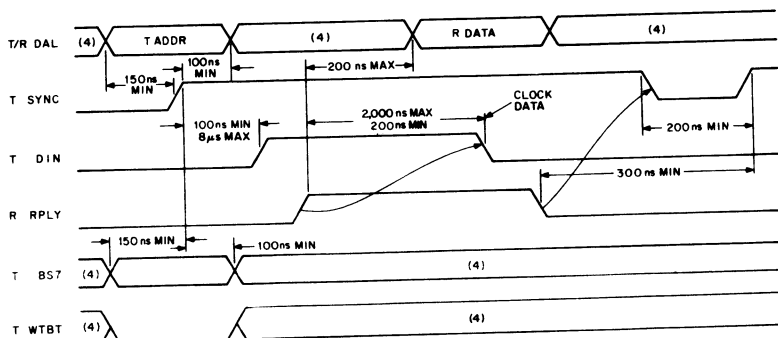
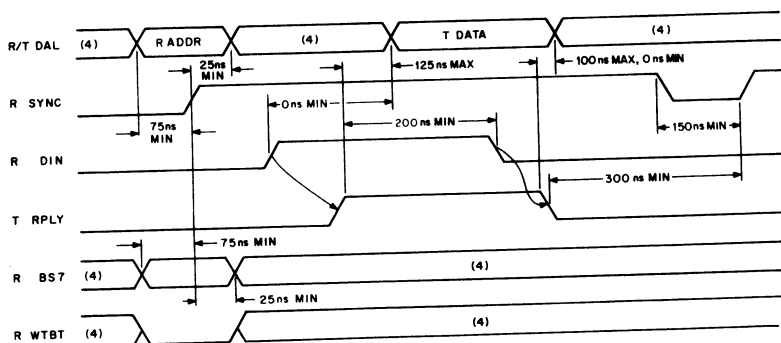


Figure 9-1 DATI Bus Cycle



TIMING AT MASTER DEVICE



TIMING AT SLAVE DEVICE

NOTES:

1. Timing shown at Master and Slave Device  
Bus Driver inputs and Bus Receiver Outputs.
2. Signal name prefixes are defined below:  
T = Bus Driver Input  
R = Bus Receiver Output
3. Bus Driver Output and Bus Receiver Input  
signal names include a "B" prefix.
4. Don't care condition.

Figure 9-2 DATI Bus Cycle Timing

**DATO(B)** — DATO(B), illustrated in Figure 9-3, is a write operation. Data are transferred in 16-bit words (DATO) or 8-bit bytes (DATOB) from the bus master to the slave device. The data transfer output can occur after the addressing portion of a bus cycle when BWTBT L has been asserted by the bus master, or immediately following an input transfer part of a DATIO(B) bus cycle.

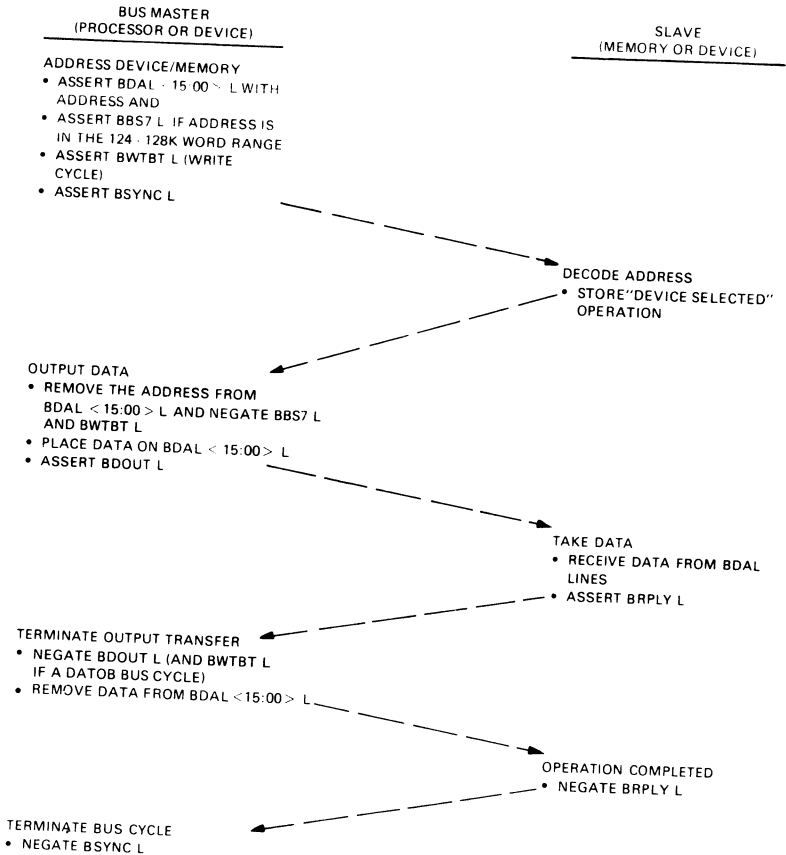


Figure 9-3 DATO or DATOB Bus Cycle

The data transfer portion of a DATO(B) bus cycle comprises a data setup and deskew time and a data hold and deskew time.

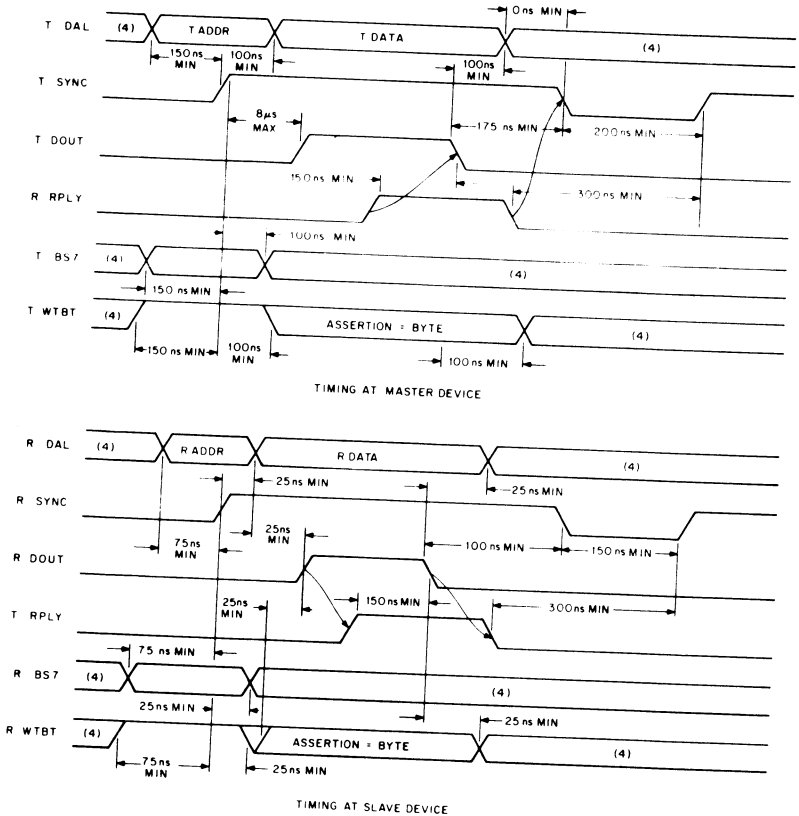
During the data setup and deskew time, the bus master outputs the data on  $BDAL<15:00> L$  at least 100 ns after  $BSYNC L$  is asserted if the transfer is a word transfer. If it is a word transfer, the bus master negates  $BWTBT L$  at least 100 ns after  $BSYNC L$  assertion.  $BWTBT L$  remains negated for the length of the bus cycle. If the transfer is a byte transfer,  $BWTBT L$  remains asserted. If it is the output of a DATIOB,  $BTWBT L$  becomes asserted and lasts the duration of the bus cycle. During a byte transfer,  $BDAL <00> L$  selects the high or low byte.

This occurs while in the addressing portion of the cycle. If asserted, the high byte ( $BDAL<15:08> L$ ) is selected; otherwise, the low byte ( $BDAL<07:00> L$ ) is selected. An asserted  $BDAL16 L$  at this time will force a parity error to be written into memory if the memory is a parity-type memory.  $BDAL17 L$  is not used for write operations. The bus master asserts  $BDOUT L$  at least 100 ns after  $BDAL$  and  $BWTBT L$  bus drivers are stable. The slave device responds by asserting  $BRPLY L$  within 10 microseconds to avoid bus time-out. This completes the data setup and deskew time.

During the data hold and deskew time the bus master receives  $BRPLY L$  and negates  $BDOUT L$ .  $BDOUT L$  must remain asserted for at least 150 ns from the receipt of  $BRPLY L$  before being negated by the bus master.  $BDAL<17:00> L$  bus drivers remain asserted for at least 100 ns after  $BDOUT L$  negation. The bus master then negates  $BDAL$  inputs.

During this time, the slave device senses  $BDOUT L$  negation. The data are accepted and the slave device negates  $BRPLY L$ . The bus master responds by negating  $BSYNC L$ . However, the processor will not negate  $BSYNC L$  for at least 175 ns after negating  $BDOUT L$ . This completes the DATO(B) bus cycle. Before the next cycle  $BSYNC L$  must remain unasserted for at least 200 ns. Figure 9-4 illustrates DATO(B) bus cycle timing.

**DATIO(B)** — The protocol for a DATIO(B) bus cycle is identical to the addressing and data transfer portions of the DATI and DATO(B) bus cycles, and is illustrated in Figure 9-5. After addressing the device, a DATI cycle is performed as explained earlier; however,  $BSYNC L$  is not negated.  $BSYNC L$  remains active for an output word or byte transfer [DATO(B)]. The bus master maintains at least 200 ns between  $BRPLY L$  negation during the DATI cycle and  $BDOUT L$  assertion. The cycle is terminated when the bus master negates  $BSYNC L$ , as described for DATO(B). Figure 9-6 illustrates DATIO(B) bus cycle timing.



NOTES

- 1 Timing shown at Master and Slave Device  
Bus Driver Inputs and Bus Receiver Outputs
- 2 Signal name prefixes are defined below  
T = Bus Driver Input  
R = Bus Receiver Output
- 3 Bus Driver Output and Bus Receiver Input  
signal names include a "B" prefix
- 4 Don't care condition

Figure 9-4 DATO or DATOB Bus Cycle Timing

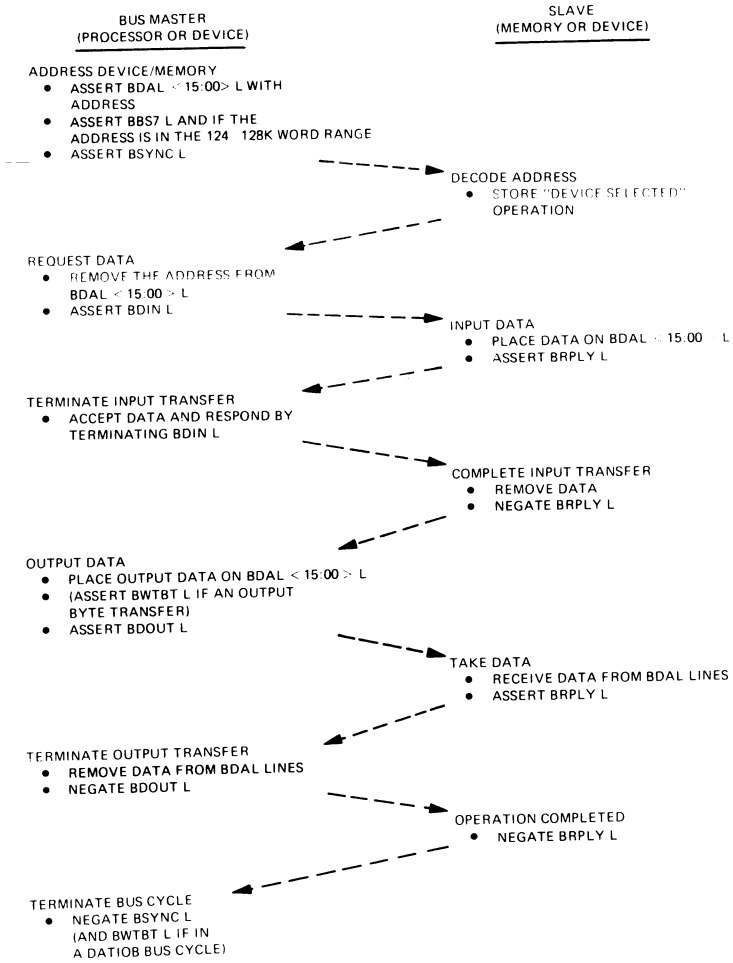
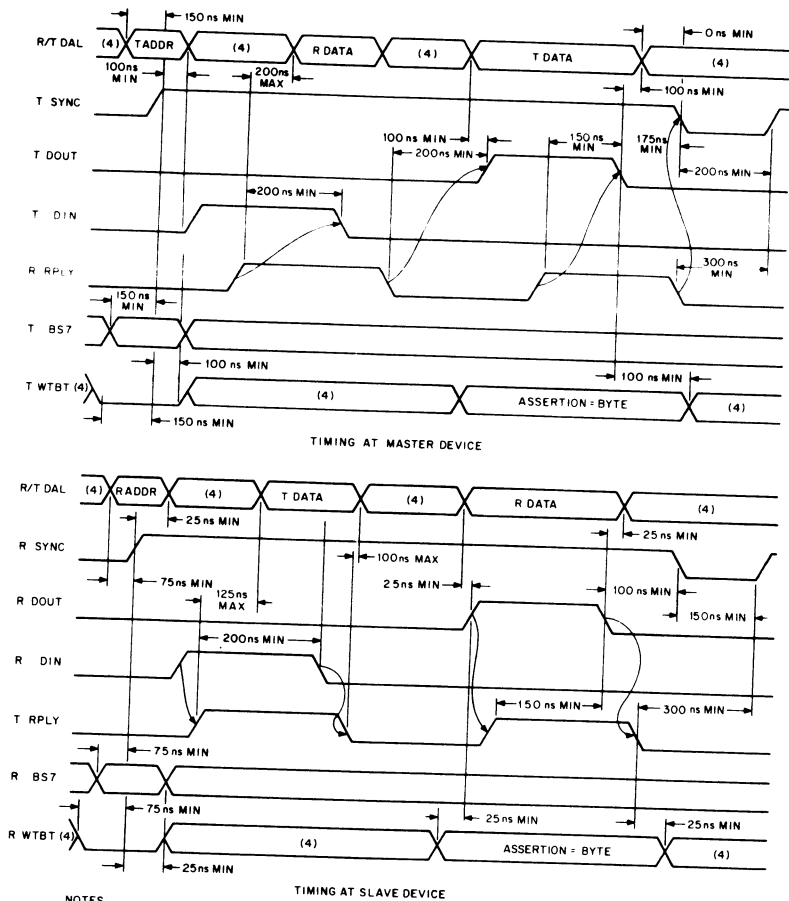


Figure 9-5 DATIO or DATIOB Bus Cycle



- NOTES
1. Timing shown at Requesting Device  
Bus Driver Inputs and Bus Receiver Outputs.
  2. Signal name prefixes are defined below  
T = Bus Driver Input  
R = Bus Receiver Output
  3. Bus Driver Output and Bus Receiver Input  
signal names include a "B" prefix.
  4. Don't care condition

Figure 9-6 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 needs to know only the starting address in memory, the starting address in mass storage, the length of the transfer, and whether the operation is read or write. When this information is available, the DMA device can transfer data directly to or from memory. Since most DMA devices must perform data transfers in rapid succession or lose data, DMA devices are provided the highest priority.

DMA is accomplished after the processor (normally bus master) has passed bus mastership to the highest-priority DMA device that is requesting the bus. The processor arbitrates all requests and grants the bus to the DMA device located electrically closest to it. A DMA device remains bus master indefinitely until it relinquishes its mastership. The following control signals are used during bus arbitration.

BDMGI L	DMA Grant Input
BDMGO L	DMA Grant Output
BDMR L	DMA Request Line
BSACK L	Bus Grant Acknowledge

**DMA Protocol**

A DMA transaction can be divided into three phases:

- Bus mastership acquisition phase
- Data transfer phase
- Bus mastership relinquish phase

During the bus mastership acquisition phase, a DMA device requests the bus by asserting BDMR L. The processor arbitrates the request and initiates the transfer of bus mastership by asserting BDMGO L. The maximum time between BDMR L assertion and BDMGO L assertion is DMA latency. This time is processor-dependent. BDMGO L/BDMGI L is one signal that is daisy-chained through each module in the backplane. It is driven out of the processor on the BDMGO L pin, enters each module on the BDMGI L pin, and exits on the BDMGO L pin. This signal passes through the modules in descending order of priority until it is stopped by the requesting device. The requesting device blocks the output of BMDGO L and asserts BSACK L. If BDMR L is continuously asserted, the bus will be hung.

During the data transfer phase, the DMA device continues asserting BSACK L. The actual data transfer is performed as described earlier.



The DMA device can assert BSYNC L for a data transfer 250 ns (minimum) after it receives BDMGI L and its BSYNC L bus receiver becomes negated.

During the bus mastership relinquish phase, the DMA device relinquishes the bus by negating BSACK L. This occurs after completing (or aborting) the last data transfer cycle (BRPLY L negated). BSACK L may be negated up to a maximum of 300 ns before negating BSYNC L. Figure 9-7 illustrates the DMA protocol and Figure 9-8 illustrates DMA request/grant timing.

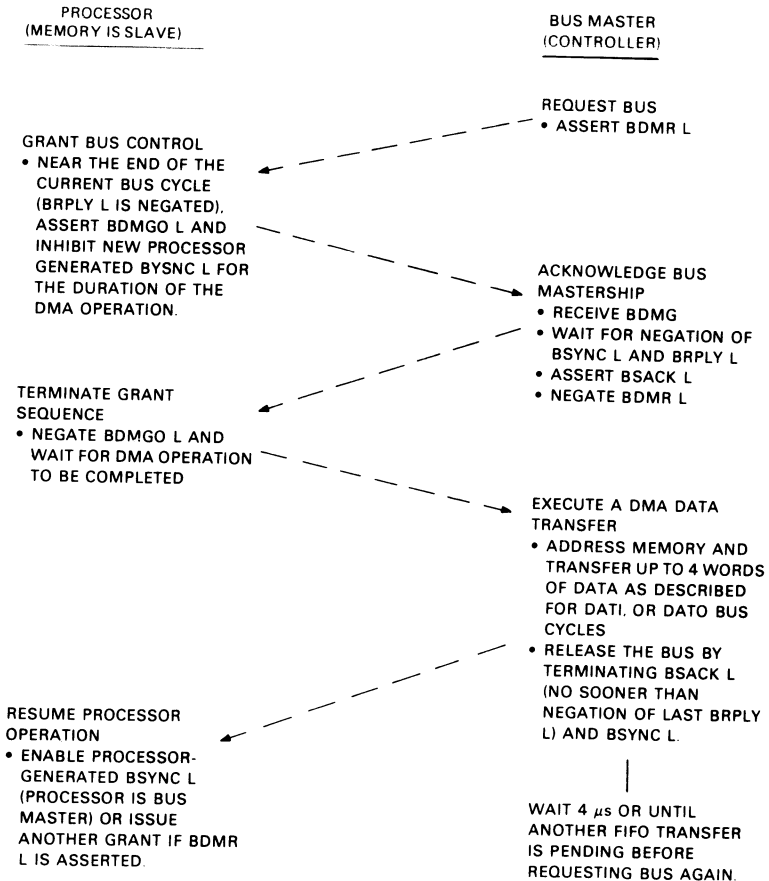


Figure 9-7 DMA Protocol

**NOTE**

If multiple data transfers are performed during this phase, consideration must be given to the use of the bus for other system functions, such as memory refresh (if required).

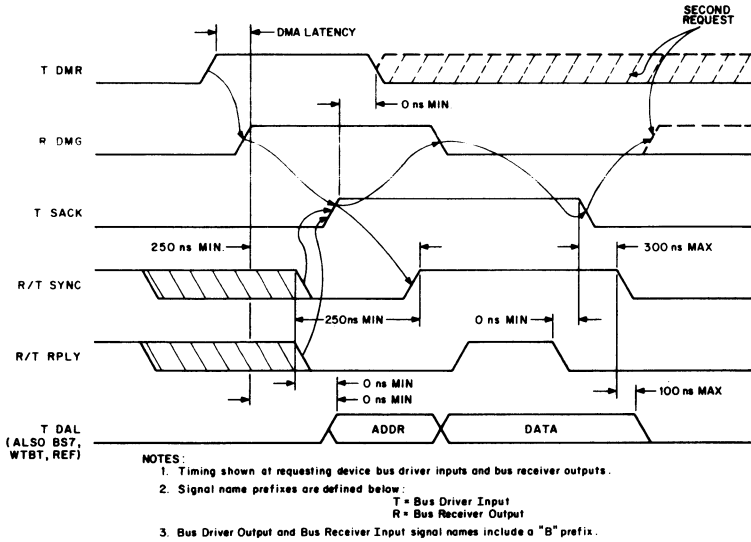


Figure 9-8 DMA Request/Grant Timing

**INTERRUPTS**

The interrupt capability of the LSI-11 Bus allows any I/O device to temporarily suspend (interrupt) current program execution and divert processor operation to service the requesting device. The processor inputs a vector from the device to start the service routine (handler). Like the device register address, hardware fixes the device vector at locations within a designated range below location 001000. The vector indicates the first of a pair of addresses. The content of the first address is read by the processor and is the starting address of the interrupt handler. The content of the second address is a new processor status word (PS). The new PS can raise the interrupt priority level, thereby preventing lower-level interrupts from breaking into the current interrupt service routine. Control is returned to the interrupted

program when the interrupt handler is ended. The original interrupted program's address (PC) and its associated PS are stored on a stack. The original PC and PS are restored by a return from interrupt (RTI or RTT) instruction at the end of the handler. The use of the stack and the LSI-11 Bus interrupt scheme can allow interrupts to occur within interrupts (nested interrupts), depending on the PS.

Interrupts can be caused by LSI-11 Bus options or the CPU. Those interrupts that originate from within the processor are called traps. Traps are caused by programming errors, hardware errors, special instructions, and maintenance features.

The LSI-11 Bus signals used in interrupt transactions are:

BIRQ4 L	Interrupt request priority level 4
BIRQ5 L	Interrupt request priority level 5
BIRQ6 L	Interrupt request priority level 6
BIRQ7 L	Interrupt request priority level 7

BIAKI L	Interrupt acknowledge input
BIAKO L	Interrupt acknowledge output

BDAL <21:00> L	Data/address lines
----------------	--------------------

BDIN L	Data input strobe
BRPLY L	Reply

### Device Priority

The LSI-11 Bus supports the following two methods of device priority:

- Distributed Arbitration—priority levels are implemented on the hardware. When devices of equal priority level request an interrupt, priority is given to the device electrically closest to the processor.
- Position-Defined Arbitration—priority is determined solely by electrical position on the bus. The closer a device is to the processor, the higher its priority is.

### Interrupt Protocol

Interrupt protocol on the LSI-11/23 has three phases: interrupt request phase, interrupt acknowledge and priority arbitration phase, and interrupt vector transfer phase. Figure 9-9 illustrates the interrupt request/acknowledge sequence.

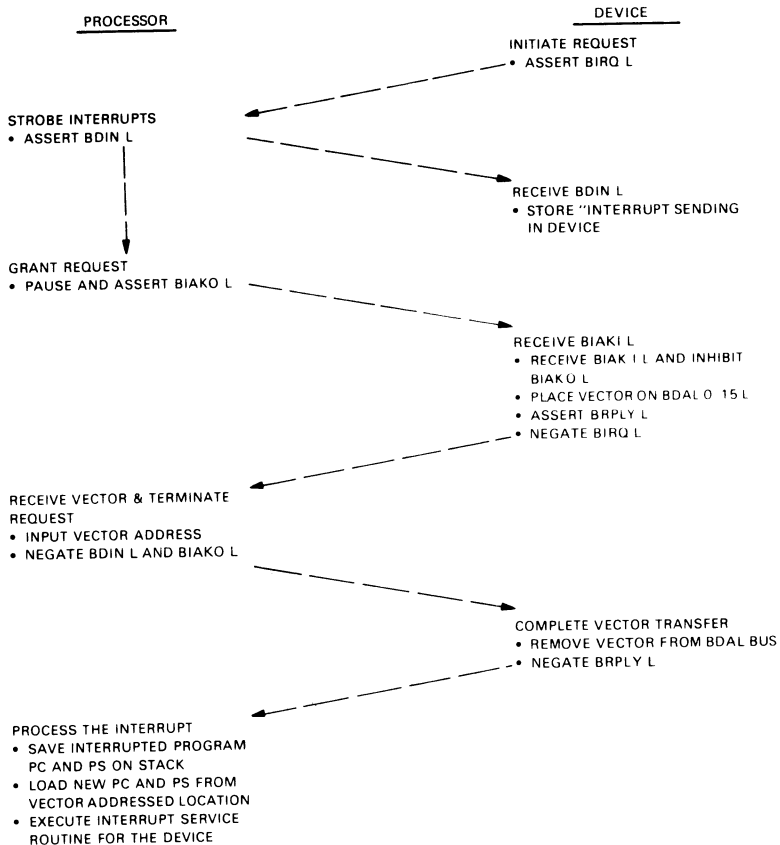


Figure 9-9 Interrupt Request/Acknowledge Sequence

The interrupt request phase begins when a device meets its specific conditions for interrupt requests. For example, the device is ready, done, or an error has occurred. The interrupt enable bit in a device status register must be set. The device then initiates the interrupt by asserting the interrupt request line(s). BIRQ4 L is the lowest hardware priority level and is asserted for all interrupt requests for compatibility with previous LSI-11 processors. The level a device is configured at must also be asserted. A special case exists for level 7 devices which

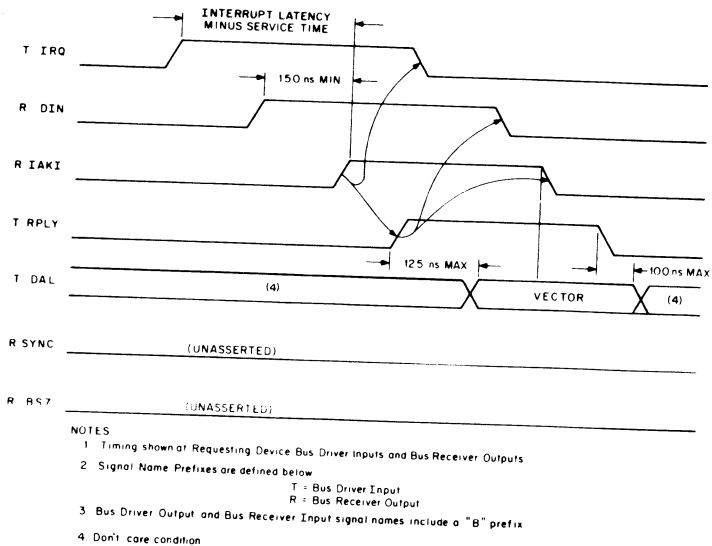


Figure 9-10 Interrupt Protocol Timing

must also assert level 6. See the arbitration discussion below involving the 4-level scheme for an explanation.

Interrupt Level	Lines Asserted by Device
4	BIRQ4 L
5	BIRQ4 L, BIRQ5 L
6	BIRQ4 L, BIRQ6 L
7	BIRQ4 L, BIRQ6 L, BIRQ7 L

The interrupt request line remains asserted until the request is acknowledged.

During the interrupt acknowledge and priority arbitration phase the LSI-11/23 processor will acknowledge interrupts under the following conditions:

1. The device interrupt priority is higher than the current PS<7:5>.
2. The processor has completed instruction execution and no additional bus cycles are pending.

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 level 4, 5, and 6 devices, since level 7 devices very seldom are necessary.

<b>Device Priority Level</b>	<b>Line(s) Monitored</b>
4	BIRQ5, BIRQ6
5	BIRQ6
6	BIRQ7
7	—

3. If no higher-level device is requesting an interrupt, the acknowledge is blocked by the device. (BIAKO L is not asserted.) Arbitration logic within the device uses the leading edge of BDIN L to clock a flip-flop that blocks BIAKO L. Arbitration is won, and the interrupt vector transfer phase begins.
4. If a higher-level request line is active, the device disqualifies itself and asserts BIAKO L to propagate the acknowledge to the next device along the bus.

Signal timing must be carefully considered when implementing 4-level interrupts. Note Figure 9-10.

If a single-level interrupt device receives the acknowledge, it reacts as follows:

- If not requesting an interrupt, the device asserts BIAKO L and the acknowledge propagates to the next device on the bus.
- If the device was requesting an interrupt, the acknowledge is blocked using the leading edge of BDIN L and arbitration is won. The interrupt vector transfer phase begins.

The interrupt vector transfer phase is enabled by BDIN L and BIAKI L. The device responds by asserting BRPLY L and its BDAL<15:00> L bus driver inputs with the vector address bits. The BDAL bus driver inputs must be stable within 125 ns (maximum) after BRPLY L is asserted. The processor then inputs the vector address and negates BDIN L and BIAKO L. The device then negates BRPLY L and 100 ns (maximum) later removes the vector address bits. The processor then enters the device's service routine.

#### NOTE

Propagation delay from BIAKI L to BIAKO L must not be greater than 500 ns per LSI-11 Bus slot.

The device must assert BRPLY L within 10 microseconds (maximum) after the processor asserts BIAKI L.

#### LSI-11/23 Four-Level Interrupt Configurations

If you have high-speed peripherals and desire better software performance, you can use the 4-level interrupt scheme. Both position-independent and position-dependent configurations can be used with the 4-level interrupt scheme.

The position-independent configuration is illustrated in Figure 9-11. This allows peripheral devices that use the 4-level interrupt scheme to be placed in the backplane in any order. These devices must send out interrupt requests and monitor higher-level request lines as described. The level 4 request is always asserted by a requesting device regardless of priority, to allow compatibility if an LSI-11 or LSI-11/2 processor is in the same system. If two or more devices of equally high priority request an interrupt, the device physically closest to the processor will win arbitration. Devices that use the single-level interrupt scheme must be modified or placed at the end of the bus for arbitration to function properly.

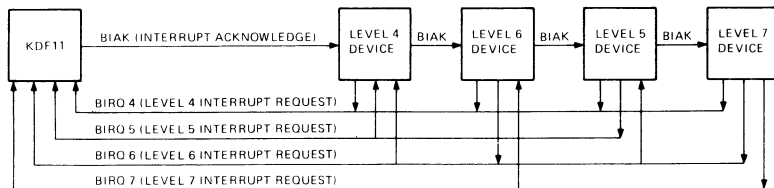


Figure 9-11 Position-Independent Configuration

The position-dependent configuration is illustrated in Figure 9-12. This configuration is simpler to implement. A constraint is that peripheral devices must be inserted with the highest-priority device located closest to the processor and the remaining devices placed in the backplane in decreasing order of priority, with the lowest-priority devices farthest from the processor. With this configuration each device has to assert only its own level and level 4 (for compatibility with an LSI-11 or LSI-11/2). Monitoring higher level request lines is unnecessary. Arbitration is achieved through the physical positioning of each device on the bus. Single-level interrupt devices on level 4 should be positioned last on the bus.

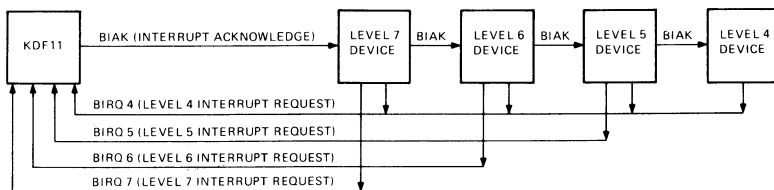


Figure 9-12 Position-Dependent Configuration

**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 addressed simultaneously. The sequence of addresses required for refreshing the memories is determined by the specific requirements for each memory. The complete memory refresh cycle consists of a series of refresh bus transactions. A new address is used for each transaction. A complete memory refresh cycle must be completed within 1 or 2 ms. Multiple data transfers by DMA devices must be avoided since they could delay memory refresh cycles.

### **Halt**

Assertion of BHALT L for at least 25  $\mu$ s interrupts the processor, which stops program execution and forces the processor unconditionally into console ODT mode.

### **Initialization**

Devices along the bus are initialized when BINIT L is asserted. The processor can assert BINIT L as a result of executing a RESET instruction 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).

**BDCOK H** — When asserted, this indicates that dc power has been stable for at least 3 ms. Once asserted, this line remains asserted until the power fails. It indicates that only 5 microseconds of dc power reserve remains.

**BPOK H** — When asserted, this indicates that there is at least an 8 ms reserve of dc power and that BDCOK H has been asserted for at least 70 ms. Once BPOK H has been asserted, it must remain asserted for at least 3 ms. The negation of this line, the first event in the power-fail sequence, indicates that power is failing and that only 4 ms of dc power reserve remains.

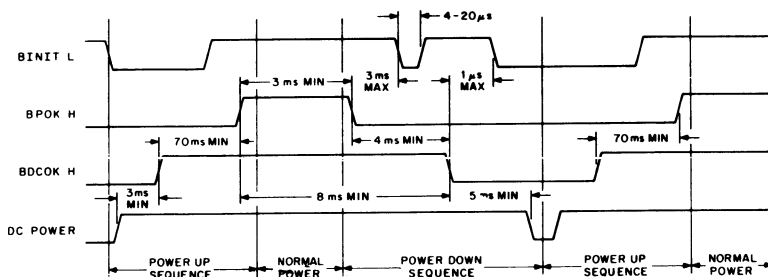
### **Power-Up/Down Protocol**

Power-up protocol begins when the power supply applies power with BDCOK H negated. This forces the processor to assert BINIT L. When the dc voltages are stable, the power supply or other external device asserts BDCOK H. The processor responds by clearing the PS, floating point status register (FPS), and floating point exception register (FEC). BINIT L is asserted for 12.6 microseconds and then negated for

110 microseconds. The processor continues to test for BPOK H until it is asserted. The power supply asserts BPOK H 70 ms (minimum) after BDCOK H is asserted. The processor then performs its power-up sequence. Normal power must be maintained at least 3.0 ms before a power-down sequence can begin. 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 at location 24<sub>8</sub>. The end of the routine is terminated with a HALT instruction to avoid any possible memory corruption as the dc voltages decay.

When the processor executes the HALT instruction, it tests the BPOK H signal. If BPOK H is negated, the processor enters the power-up sequence. It clears internal registers, generates BINIT L, and continues to check for the assertion of BPOK H. If it is asserted and dc voltages are still stable, the processor will perform the rest of the power-up sequence. Figure 9-13 illustrates power-up/power-down timing.



## NOTE

Once a power down sequence is started it must be completed before a power-up sequence is started

Figure 9-13 Power-Up/Power-Down Timing

## LSI-11 BUS ELECTRICAL CHARACTERISTICS

### Signal Level Specification

#### Input Logic Levels

TTL Logical Low:	0.8 Vdc maximum
TTL Logical High:	2.0 Vdc minimum

**Output Logic Levels**

TTL Logical Low:	0.4 Vdc maximum
TTL Logical High:	2.4 Vdc minimum

**Load Definition**

AC loads comprise the maximum capacitance allowed per signal line to ground. A unit load is defined as 9.35 pF of capacitance. DC loads are defined as maximum current allowed with a signal line driver asserted or unasserted. A unit load is defined as 105  $\mu$ A in the unasserted state.

**120 Ohm LSI-11 Bus**

The electrical conductors interconnecting the bus device slots are treated as transmission lines. A uniform transmission line, terminated in its characteristic impedance, will propagate an electrical signal without reflections. Since bus drivers, receivers, and wiring connected to the bus have finite resistance and nonzero reactance, the transmission line impedance is not uniform, and introduces distortions into pulses propagated along it. Passive components of the LSI-11 Bus (such as wiring, cabling, and etched signal conductors) are designed to have a nominal characteristic impedance of 120 ohms.

The maximum length of interconnecting cable excluding wiring within the backplane is limited to 4.88 m (16 ft.).

**Bus Drivers**

Devices driving the 120 ohm LSI-11 Bus must have open collector outputs and meet the following specifications.

*DC Specifications*

Output low voltage when sinking 70 mA of current: 0.7V maximum.

Output high leakage current when connected to 3.8 Vdc: 25  $\mu$ A (even if no power is applied, except for BDCOK H and BPOK H).

These conditions must be met at worst-case supply voltage, temperature, and input signal levels.

*AC Specifications*

Bus driver output pin capacitive load: Not to exceed 10 pF.

Propagation delay: Not to exceed 35 ns.

Skew (difference in propagation time between slowest and fastest gate): Not to exceed 25 ns.

Rise/Fall Times: Transition time (from 10% to 90% for positive transition, and from 90% to 10% for negative transition) must be no faster than 10 ns.

### Bus Receivers

Devices that receive signals from the 120 ohm LSI-11 Bus must meet the following requirements.

#### DC Specifications

Input low voltage (maximum): 1.3V.

Input high voltage (minimum): 1.7V.

Maximum input current when connected to 3.8 Vdc: 80  $\mu$ A even if no power is applied.

These specifications must be met at worst-case supply voltage, temperature, and output signal conditions

#### AC Specifications

Bus receiver input pin capacitance load:  
Not to exceed 10 pF.

Propagation delay: Not to exceed 35 ns.

Skew (difference in propagation time between slowest and fastest gate): Not to exceed 25 ns.

### Bus Termination

The 120 ohm LSI-11 Bus must be terminated at each end by an appropriate terminator, as illustrated in Figure 9-14. This is to be done as a voltage divider with its Thevenin equivalent equal to 120 ohms and 3.4V nominal. This type of termination is provided by an REV11-A refresh/boot/terminator, BDV11-AA, KPU11-B, TEV11, or in the case of 22-bit systems, by the H9275 backplane, itself.

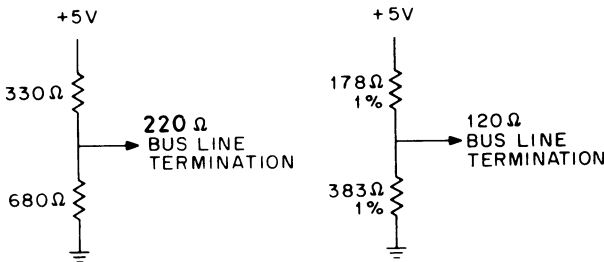


Figure 9-14 Bus Line Terminations

Each of the several LSI-11 Bus lines (all signals whose mnemonics start with the letter B) must see an equivalent network with the following characteristics at each end of the bus:

Input impedance (with respect to ground)	120 ohm +5%, -15%
Open circuit voltage	3.4 Vdc +5%
Capacitance Load	Not to exceed 30 pF

#### NOTE

The resistive termination may be provided by the combination of two modules (i.e., the processor module supplies 220 ohms to ground). Both of these terminators must be physically resident within the same backplane.

#### Bus Interconnecting Wiring

**Backplane Wiring** — The wiring that connects all device interface slots on the LSI-11 must meet the following specifications:

1. The conductors must be arranged such that each line exhibits a characteristic impedance of 120 ohms (measured with respect to the bus common return).
2. Crosstalk between any two lines must be no greater than 5%. Note that worst-case crosstalk is manifested by simultaneously driving all but one signal line and measuring the effect on the undriven line.
3. DC resistance of the signal path, as measured between the near-end terminator and the far-end terminator module (including all intervening connectors, cables, backplane wiring, connector-module etch, etc.) must not exceed 2 ohms.
4. DC resistance of common return path, as measured between the near-end terminator and the far-end terminator module (including all intervening connectors, cables, backplane wiring, connector-module etch, etc.) must not exceed an equivalent of 2 ohms per signal path. Thus, the composite signal return path dc resistance must not exceed 2 ohms divided by 40 bus lines, or 50 milliohms. Note that although this common return path is nominally at ground potential, the conductance must be part of the bus wiring. The specified low-impedance return path must be provided by the bus wiring as distinguished from the common system or power ground path.

**Intra-Backplane Bus Wiring** — The wiring that connects the bus connector slots within one contiguous backplane is part of the overall bus transmission line. Owing to implementation constraints, the nominal characteristic impedance of 120 ohms may not be achievable. Distributed wiring capacitance in excess of the amount required to achieve the nominal 120 ohm impedance may not exceed 60 pF per signal line per backplane.

**Power and Ground** — Each bus interface slot has connector pins assigned for the following dc voltages. The maximum allowable current per pin is 1.5 A. +5 Vdc must be regulated to  $\pm 5\%$  with a maximum ripple of 100 mV pp. +12 Vdc must be regulated to  $\pm 3\%$  with a maximum ripple of 200 mV pp.

- +5Vdc—Three pins (4.5 A maximum per bus device slot)
- +12 Vdc—Two pins (3.0 A maximum per bus device slot)
- Ground—Eight pins (shared by power return and signal return)

**NOTE**

Power is not bused between backplanes on any interconnecting bus cables.

**SYSTEM CONFIGURATIONS**

LSI-11 Bus systems can be divided into two types:

1. Systems containing one backplane
2. Systems containing multiple backplanes

Before configuring any system, three characteristics for each module in the system must be known. These characteristics are:

- Power consumption—+5 Vdc and +12 Vdc current requirements.
- AC bus loading—the amount of capacitance a module presents to a bus signal line. AC loading is expressed in terms of ac loads where one ac load equals 9.35 pF of capacitance.
- DC bus loading—the amount of dc leakage current a module presents to a bus signal when the line is high (undriven). DC loading is expressed in terms of dc loads where one dc load equals 210 microamperes (nominal).

Power consumption, ac loading, and dc loading specifications for each module are included in the *Microcomputer Interface Handbook*.

**NOTE**

The ac and dc loads and the power consumption of the processor module, terminator module, and backplane must be included in determining the total loading of a backplane.

### Rules for Configuring Single Backplane Systems

- As illustrated in Figure 9-15, 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.

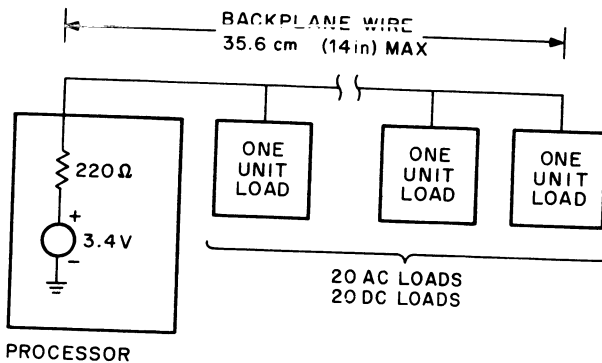


Figure 9-15 Single Backplane Configuration

### Rules for Configuring Multiple Backplane Systems

- As illustrated in Figure 9-16, 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 cable(s) connecting the first two backplanes are 61 cm (2 ft ) or greater in length.
- The cable(s) connecting the second backplane to the third backplane are 122 cm (4 ft ) longer or shorter than the cable(s) connecting the first and second backplanes.
- The combined length of both cables cannot exceed 4.88 m (16 ft ).
- The cables used must have a characteristic impedance of 120 ohms.

### **Power Supply Loading**

Total power requirements for each backplane can be determined by obtaining the total power requirements for each module in the backplane. Obtain separate totals for +5V and +12V power. Power requirements for each module are specified in the *Microcomputer 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 shows the bus pins, mnemonics and descriptions. It is defined by processor so that differences, when they occur, can be seen easily.

### **MODULE CONTACT FINGER IDENTIFICATION**

DIGITAL plug-in modules, including the KDF11-AC, all use the same contact finger (pin) identification system. The LSI-11 Bus is based on the use of double-height modules that plug into a 2-slot bus connector. Each slot contains 36 lines (18 each on component and solder sides of circuit board).

Slots, shown as row A and row B in Figure 9-17, 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



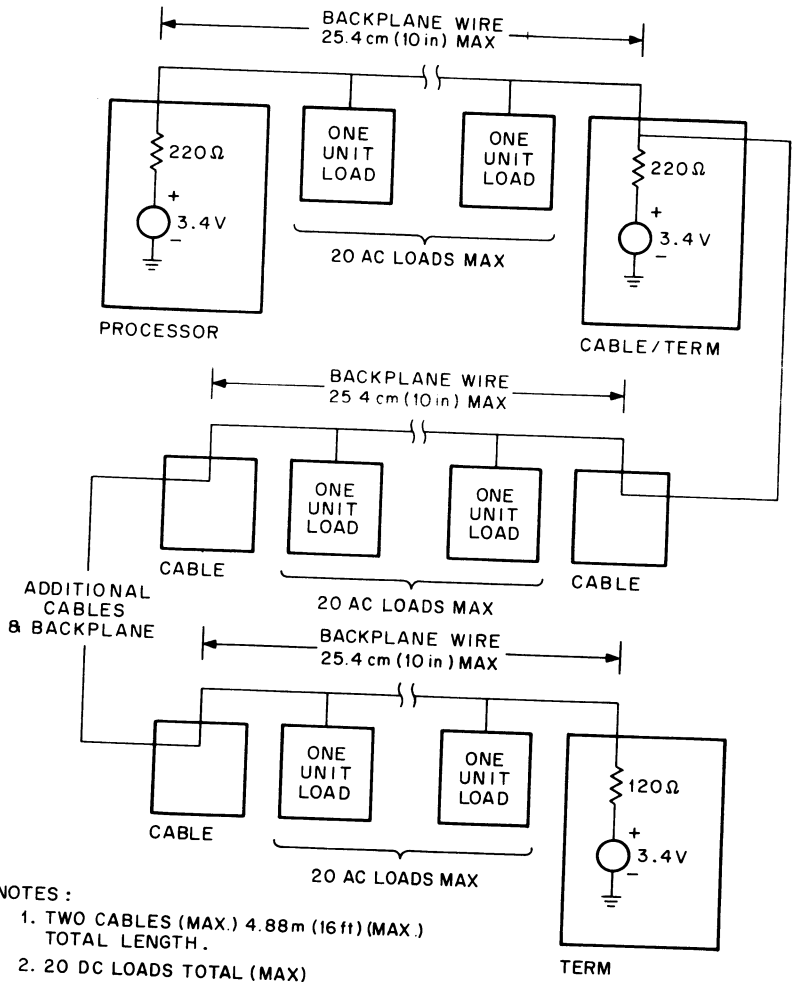


Figure 9-16 Multiple Backplane Configuration

of a slot. Table 9-4 lists and identifies the bus pins of the double-height module. The bus pin identifier ending with a 1 is found on the component side of the board, while a bus pin identifier ending with a 2 is found on the solder side of the board. A typical pin is designated as follows.

BE2

Slot (Row) Identifier  
"Slot B"

Module Side Identifier  
"Side 2" (solder side)

Pin Identifier  
"Pin E"

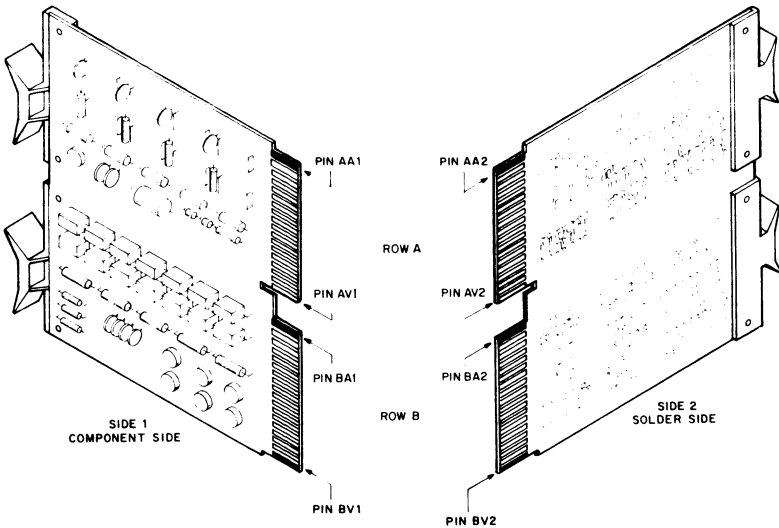


Figure 9-17 Double-Height Module Contact Finger Identification

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

Table 9-4 Bus Pin Identifiers

BUS PIN	MNEMONICS	DESCRIPTION
AA1	BIRQ5 L	Interrupt Request Priority Level 5
AB1	BIRQ6 L	Interrupt Request Priority Level 6

<b>BUS PIN</b>	<b>MNEMONICS</b>	<b>DESCRIPTION</b>
AC1	BDAL16 L	Extended address bit during addressing protocol; memory error data line during data transfer protocol.
AD1	BDAL17 L	Extended address bit during addressing protocol; memory error logic enable during data transfer protocol.
AE1	SSPARE1 (Alternate +5B)	Special Spare—not assigned or bused in DIGITAL cable or backplane assemblies; available for user connection. Optionally, this pin may be used for +5V battery (+5B) backup power to keep critical circuits alive during power failures. A jumper is required on LSI-11 Bus options to open (disconnect) the +5B circuit in systems that use this line as SSPARE1.
AF1	SSPARE2	Special Spare—not assigned or bused in DIGITAL cable or backplane assemblies; available for user interconnection. In the highest-priority device slot, the processor may use this pin for a signal to indicate its RUN state.
AH1	SSPARE 3 SRUN simultaneously	Special Spare—not assigned or bused in DIGITAL cable or backplane assemblies; available for user interconnection. An alternate SRUN signal may be connected in the highest-priority set.
AJ1	GND	Ground—System signal ground and dc return.

<b>BUS PIN</b>	<b>MNEMONICS</b>	<b>DESCRIPTION</b>
AK1	MSPAREA	Maintenance Spare—Normally connected together on the backplane at each option location (not bused connection).
AL1	MSPAREB	Maintenance Spare—Normally connected together on the backplane at each option location (not bused connection).
AM1	GND	Ground—System signal ground and dc return.
AN1	BDMR L	Direct Memory Access (DMA) Request—A device asserts this signal to request bus master-ship. The processor arbitrates bus mastership between itself and all DMA devices on the bus. If the processor is not bus master (it has completed a bus cycle and BSYNC L is not being asserted by the processor), it grants bus mastership to the requesting device by asserting BDMGO L. The device responds by negating BDMR L and asserting BSACK L.
AP1	BHALT L	Processor Halt—When BHALT L is asserted for at least 25 $\mu$ s, the processor services the halt interrupt and responds by halting normal program execution. External interrupts are ignored but memory refresh interrupts in LSI-11 are enabled if W4 on M7264 and M7264-YA processor modules is removed and DMA request/grant sequences are enabled. The processor executes the ODT microcode and the console device operation is invoked.

BUS PIN	MNEMONICS	DESCRIPTION
AR1	BREF L	Memory Refresh—Asserted by a DMA device. This signal forces all dynamic MOS memory units requiring bus refresh signals to be activated for each BSYNC L/BDIN L bus transaction.
		<b>CAUTION</b>
		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.
AS1	+12B or +5B	* +12 Vdc or +5V battery backup power to keep critical circuits alive during power failures. This signal is not bused to BS1 in all DIGITAL backplanes. A jumper is required on all LSI-11 Bus options to open (disconnect) the backup circuit from the bus in systems that use this line at the alternate voltage.
AT1	GND	Ground—System signal ground and dc return.
AU1	PSPARE 1	Spare (Not assigned. Customer usage not recommended.) Prevents damage when modules are inserted upside down.
AV1	+5B	* +5V Battery Power—Secondary +5V power connection. Battery power can be used with certain devices.
BA1	BDCOK H	DC Power OK—Power supply-generated signal that is asserted when there is sufficient dc voltage available to sustain reliable system operation.

<b>BUS PIN</b>	<b>MNEMONICS</b>	<b>DESCRIPTION</b>
BB1	BPOK H	Power OK—Asserted by the power supply 70 ms after BDCOK negated when ac power drop below the value required to sustain power (approximately 75% of nominal). When negated during processor operation, a power-fail trap sequence is initiated.
BC1	SSPARE4 BDAL 18L (on Q22 only)	Special Spare in Q bus—Not assigned. Bussed in H9275 and H9276. Cable and backplane assemblies; available for use in interconnection.
BD1	SSPARE5 BDAL 19L (on Q22 only)	Caution. These pins may be used as test points by DIGITAL in some options.
BE1	SSPARE6 BDAL 20L	In Q22 these bussed address lines are Address Lines <21: 18> currently not used during data time.
BF1	SSPARE7 BDAL 21L	In Q22 these bussed address lines are Address Lines <21: 18> currently not used during data time.
BH1	SSPARE8	Special Spare—Not assigned or used in DIGITAL cable and backplane assemblies; available for user interconnection.
BJ1	GND	Ground—System signal ground and dc return.
BK1 BL1	MSPAREB MSPAREB	Maintenance Spare—Normally connected together on the backplane at each option location (not a bused connection).
BM1	GND	Ground—System signal ground and dc return.

<b>BUS PIN</b>	<b>MNEMONICS</b>	<b>DESCRIPTION</b>
BN1	BSACK L	This signal is asserted by a DMA device in response to the processor's BDMGO L signal, indicating that the DMA device is bus master.
BP1	BIRQ7 L	Interrupt request priority level 7
BR1	BEVNT L	External Event Interrupt Request—When asserted, the processor responds (if PS bit 7 is 0) by entering a service routine via vector address 100 <sub>8</sub> . A typical use of this signal is a line time clock interrupt.
BS1	+12B	* +12 Vdc battery backup power (not bused to AS1 in all DIGITAL backplanes).
BT1	GND	Ground—System signal ground and dc return.
BU1	PSPARE2	Power Spare 2 (not assigned a function, not recommended for use). If a module is using -12V (on pin AB2) and if the module is accidentally inserted upside down in the backplane, -12 Vdc appears on pin BU1.
BV1	+5	+5V Power—Normal +5 Vdc system power.
AA2	+5	+5V Power—Normal +5 Vdc system power.
AB2	-12	* -12V Power—-12 Vdc (optional) power for devices requiring this voltage.

<b>BUS PIN</b>	<b>MNEMONICS</b>	<b>DESCRIPTION</b>
		<b>NOTE</b>
		LSI-11 modules which require negative voltages contain an inverter circuit (on each module) which generates the required voltage(s). Hence, -12V power is not required with DIGITAL-supplied options.
AC2	GND	Ground—System signal ground and dc return.
AD2	+12	+12V Power—12 Vdc system power.
AE2	BDOUT L	Data Output—BDOUT, when asserted, implies that valid data is available on BDAL <0:15> L and that an output transfer, with respect to the bus master device, is taking place. BDOU L is deskewed with respect to data on the bus. The slave device responding to the BDOU L signal must assert BRPLY L to complete the transfer.
AF2	BRPLY L	Reply—BRPLY L is asserted in response to BDIN L or BDOU L and during IAK transactions. It is generated by a slave device to indicate that it has placed its data on the BDAL bus or that it has accepted output data from the bus.
AH2	BDIN L	Data Input—BDIN L is used for two types of bus operation:  When asserted during BSYNC L time, BDIN L implies an input transfer with respect to the current bus master, and requires a response (BRPLY L). BDIN L is asserted when the master de-



BUS PIN	MNEMONICS	DESCRIPTION
AJ2	BSYNC L	<p>vice is ready to accept data from a slave device.</p> <p>When asserted without BSYNC L, it indicates that an interrupt operation is occurring.</p> <p>The master device must deskew input data from BRPLY L.</p>
AK2	BWTBT L	<p>Synchronize—BSYNC L is asserted by the bus master device to indicate that it has placed an address on BDAL &lt;0:17&gt; L. The transfer is in process until BSYNC L is negated.</p> <p>Write/Byte—BWTBT L is used in two ways to control a bus cycle:</p> <p>It is asserted at the leading edge of BSYNC L to indicate that an output sequence is to follow (DATO or DATOB), rather than an input sequence.</p> <p>It is asserted during BDOUT L, in a DATOB bus cycle, for byte addressing.</p>
AL2	BIRQ4 L	<p>Interrupt Request Priority Level 4— A level 4 device asserts this signal when its interrupt enable and interrupt request flips-flops are set. If the PS word bit 7 is 0, the processor responds by acknowledging the request by asserting BDIN L and BIAKO L.</p>
AM2 AN2	BIAKI L BIAKO L	<p>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</p>

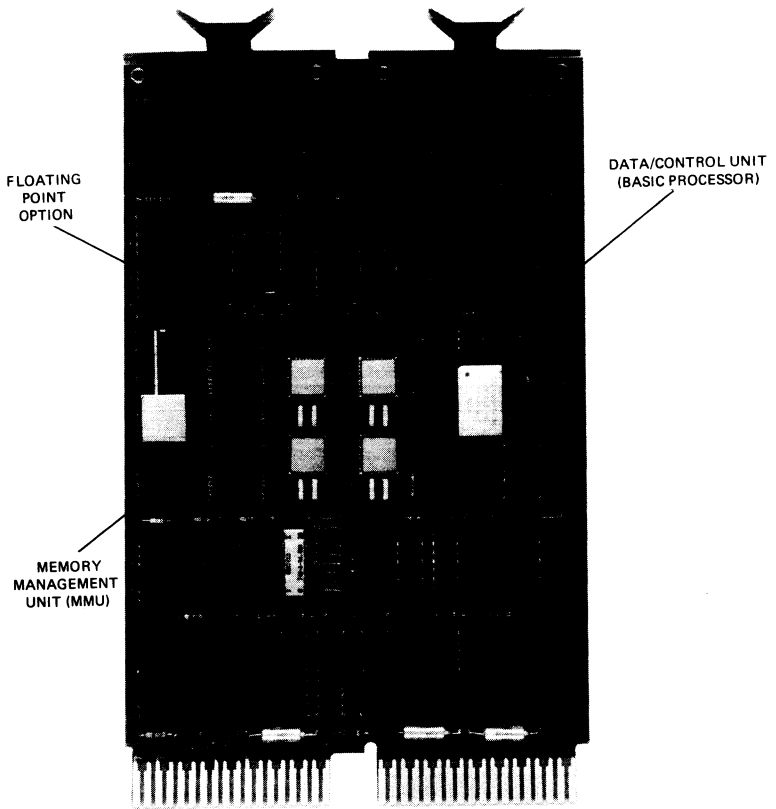
<b>BUS PIN</b>	<b>MNEMONICS</b>	<b>DESCRIPTION</b>
		<p>device electrically closest to the processor. This device accepts the interrupt acknowledge under two conditions:</p> <p>1) The device requested the bus by asserting BIRQXL, and 2) the device has the highest-priority interrupt request on the bus at that time.</p> <p>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.</p>
AP2	BBS7 L	<p>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 &lt;0:12&gt; L when BBS7 L is asserted is the address within the I/O page.</p>
AR2 AS2	BDMGI L BDMGO L	<p>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,</p>

BUS PIN	MNEMONICS	DESCRIPTION
		<p>the device passes the grant (asserts BDMGO L) to the next device on the bus. This process continues until the requesting device acknowledges the grant.</p> <p><b>CAUTION</b> DMA device transfers must not interfere with the memory refresh cycle.</p>
AT2	BINIT L	<p>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.</p>
AU2 AV2	BDALO L BDAL1 L	<p>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.</p>
BA2	+5	<p>+5V Power—Normal +5 Vdc system power.</p>
BB2	-12	<p>* -12V Power—-12 Vdc (optional) power for devices requiring this voltage.</p>
BC2	GND	<p>Ground—System signal ground and dc return.</p>

<b>BUS PIN</b>	<b>MNEMONICS</b>	<b>DESCRIPTION</b>
BD2	+12	+12V Power—+12V system power.
BE2	BDAL2 L	Data/Address Lines—These 14 lines are part of the 16-line data/address bus previously described.
BF2	BDAL3 L	
BH2	BDAL4 L	
BJ2	BDAL5 L	
BK2	BDAL6 L	
BL2	BDAL7 L	
BM2	BDAL8 L	
BN2	BDAL9 L	
BP2	BDAL10 L	
BR2	BDAL11 L	
BS2	BDAL12 L	
BT2	BDAL13 L	
BU2	BDAL14 L	
BV2	BDAL15 L	

\* Voltages normally not supplied by DIGITAL.





## CHAPTER 10

# MEMORY MANAGEMENT

The LSI-11/23 processor implements a 256 KB or a 4 megabyte physical address space. This improves the 64 KB maximum physical address space previously available in LSI-11 processors. The mapping or translation of 16-bit virtual addresses to 18-bit or 22-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 the PDP-11/34, PDP-11/60, and PDP-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 or 22-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.

### Basic Addressing

The PDP-11 family word length is 16 bits; however, the LSI-11 Bus and the LSI-11/23 addressing logic are 22 bits wide. While a 16-bit word can generate virtual address references up to 32K words (64 KB), the LSI-11/23 and the LSI-11 Bus can reference 22-bit physical addresses up to 4096 KB (2048 K Words). The extra bits of addressing logic provide the basic framework for expanding memory references.

### NOTE

In the case of the LSI-11/23 and PDP-11/23-PLUS 22-bit addresses are generated and presented directly on the LSI-11 Bus directly by the processor.

8 KB of address space are reserved for the I/O device registers. Thus 4088 KB remain for the main memory.

### Active Page Registers

The memory management unit uses two sets of eight 32-bit active page registers (APRs). 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. The active page registers are illustrated in Figure 10-1.

### Multiple Address Space

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 time-sharing systems functionally between the kernel and user modes, a minimum of space control housekeeping is required as the multitasking 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.



**Capabilities Provided by Memory Management**

Memory Size:	4096 KB (4 megabytes plus 4088 KB for the I/O Page)
Address Space:	Virtual 64 KB (16 bits) Physical 256 KB (18 bits), or 4096 KB (22-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

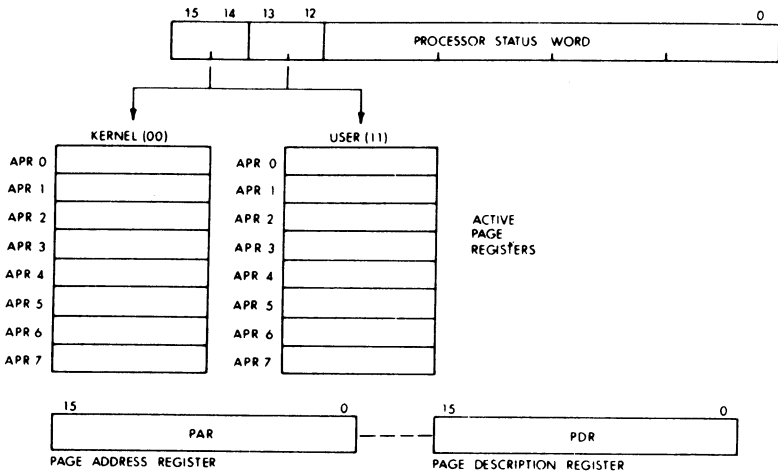


Figure 10-1 Active Page Registers

**MEMORY RELOCATION**

When the memory management unit is operating, the normal 16-bit address within a program 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 or 22-bit physical address. The information contained in the virtual address is combined with relocation and description information contained in an active page register to yield an 18-bit or 22-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 8 KB pages. Each page is relocated separately. This feature is useful in multitasking systems as it permits programs 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 64 KB 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.

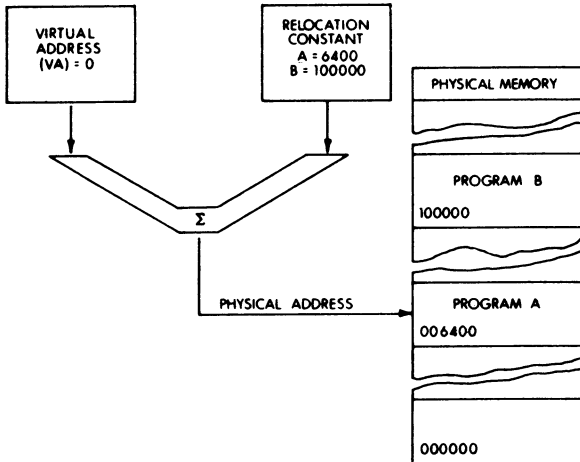


Figure 10-2 Simplified Memory Relocation

### Program Relocation

The page address registers are used to determine the starting physical address of each relocated program in physical memory. Figure 10-2 illustrates a simplified example of the relocation concept.

Program A, starting at virtual 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 consisting of from 1 to 128 blocks. Each block is 64 bytes in length. Thus, the maximum length of a page is 8,192 ( $128 \times 64$ ) 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 8 KB, only the memory actually allocated to the page may be accessed.

The relocation example in Figure 10-3 illustrates several points about memory relocation.

Although the user program appears to the processor to be in contiguous address space, the 64 KB virtual address space has been scattered throughout the physical memory space. It has been segmented, split into 8 KB 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 8 KB boundary. User page 2 has been relocated on a 4 KB boundary. Pages may be relocated to any 64 byte boundary.

Each page is relocated independently. There is no reason two or more

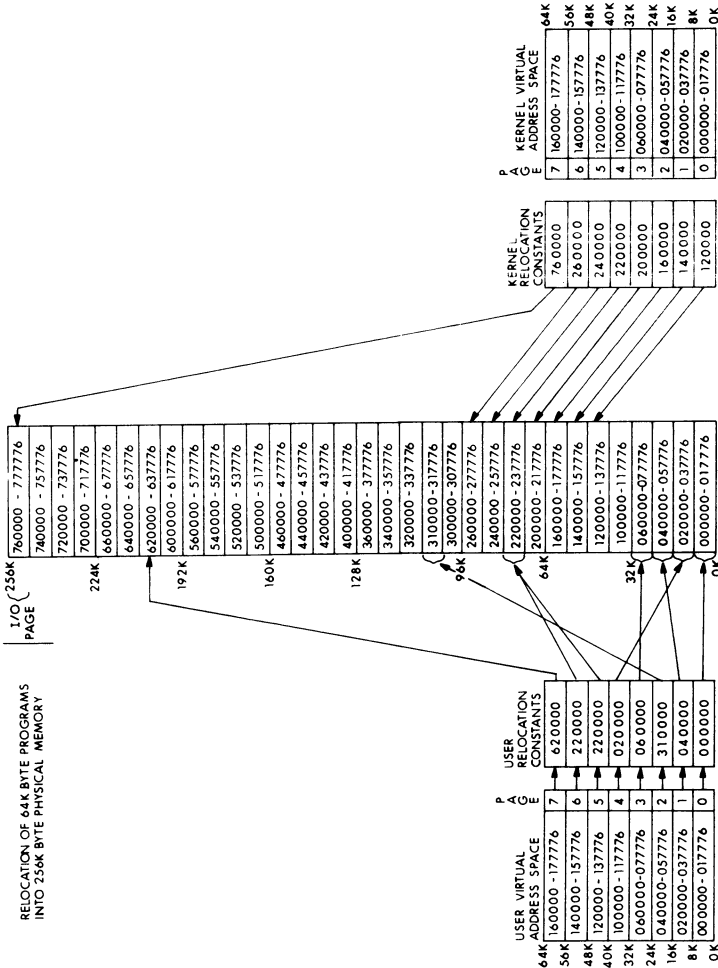


Figure 10-3 Relocation of a 64 KB Program into 256 KB Physical Memory

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 page 6 of program A is relocated to the same physical memory as user page 2 of program B. This might provide two programs access to a shared data or instructions.

The kernel virtual address space has simply been shifted up 20K words and still remains contiguous in physical memory.

The uppermost 8 KB page of physical memory is designated as the I/O page. The uppermost 8 KB of the kernel address space have 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)
Number of Pages	8 per mode
Size of Relocatable Memory	65,536 bytes, max. ( $8 \times 8,192$ )

### PROTECTION

A multiprogramming system 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 multiprogramming system, therefore, requires several types of memory protection.

- User programs must not be allowed to expand beyond their allocated space unless authorized by the operating system.
- 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 or subroutines. 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 3 (allowing complete read/write access for the operating system).

**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 communicate 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.

PS<15:14> or PS<13:12>	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. Table 10-1 lists the PSW protection fields.

**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<sub>8</sub>. 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, MTPS 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

PS Bits Condition	RTI,RTT		Traps & Interrupts		Explicit PS Access		MTPS		Power Up
	User	Kernel	User	Kernel	User	Kernel	User	Kernel	
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	Loaded From Source	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

Table 10-1 Processor Status Word Protection



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 accessed 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 LSI-11/23 copies the old PS into a temporary register and then loads the new PS value. PS<15:14> are loaded from the vector 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 content of the vector plus two locations.

**Table 10-2 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
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 16-bit page address field (PAF) that specifies the base address of the page.

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.

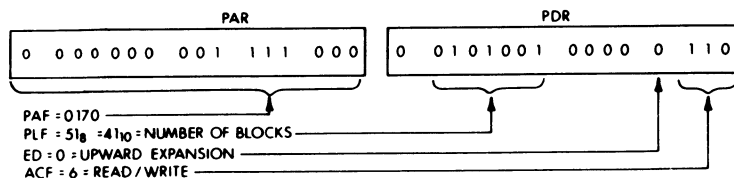


Figure 10-4 PAR, PDR

### PAGE DESCRIPTOR REGISTER (PDR)

The page descriptor register (PDR) contains information pertinent to page expansion, page length, and access control. Figure 10-4 illustrates the layout of the PAR and PDR registers. Table 10-2 lists the PAR/PDR address assignments.

### 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, by 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
01	2	Resident read-only	Abort any attempt to write into this page
10	4	Unused	Abort all accesses

ACF	Key	Description	Function
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 Figure 10-5.

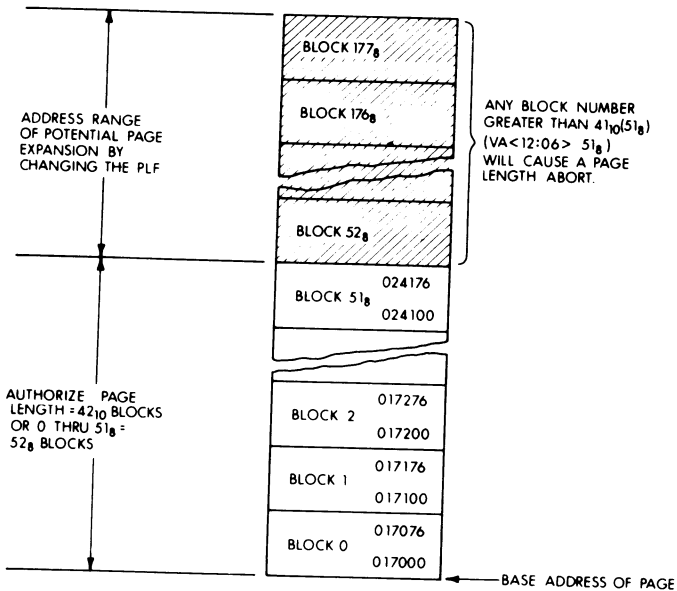


Figure 10-5 Example of an Upward-Expandable Page

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 Figure 10-6.

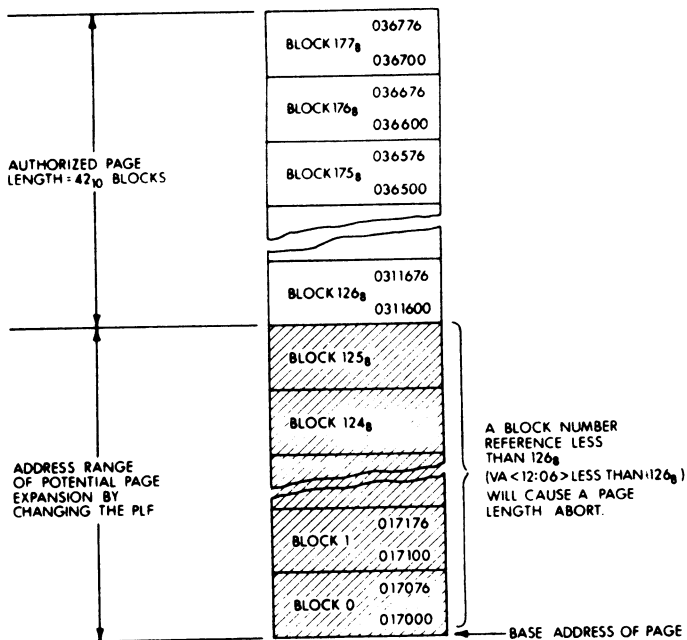


Figure 10-6 Example of a Downward-Expandable Page

### 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 overlaid 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 fault 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.

If a 42-block page is required, the PLF is derived as follows:

$$42_{10} = 52_8; 2\text{'s complement} = 126_8$$

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. Figure 10-7 illustrates the interpretation of a virtual address.

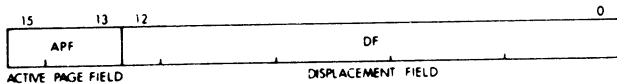


Figure 10-7 Interpretation of a Virtual Address

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 ( $2^{13} = 8 \text{ KB}$ ). The DF is further subdivided into two fields. The displacement field of the virtual address is illustrated in Figure 10-8.

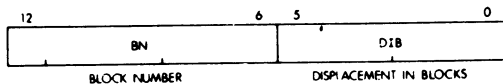


Figure 10-8 Displacement Field of Virtual Address

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 16-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 ( $3 \times 32 = 96$ ) 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 or 22-bit physical address.

### Determining the Program Physical Address

A 16-bit virtual address can specify up to 64 KB in the range from 000000 to 177776<sub>6</sub> (word boundaries are even numbers). The three most significant virtual address bits designate the PAR/PDR pair to be referenced during page address relocation. Table 10-3 lists the virtual address ranges that specify each of the PAR/PDR sets. Figure 10-9 illustrates the construction of a physical address.

**Table 10-3 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 8 KB causes unaddressable holes in the virtual address space.

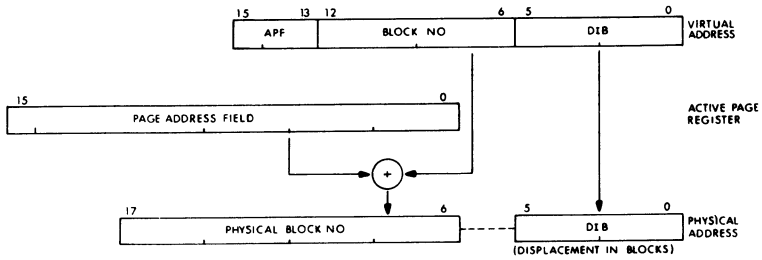


Figure 10-9 Construction of a Physical Address

### 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, illustrated in Figure 10-10, has as its address  $777572_8$ . This register is cleared by application of power or a RESET instruction.

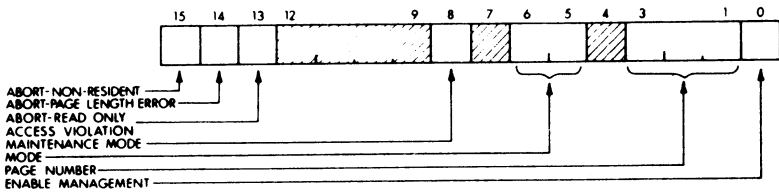


Figure 10-10 Format of Status Register 0 (SR0)

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.

**Bit: 15      Name:** Abort Nonresident

**Function:** 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.

**Bit: 14      Name:** Abort Page Length

**Function:** 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.

**Bit: 13      Name:** Abort Read-Only

**Function:** 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.

**Bit: 5-6     Name:** Mode of Operation

**Function:** Bits 5 and 6 indicate the CPU mode (user or kernel) associated with the page causing the abort (kernel = 00, user = 11).

**Bit: 1-3     Name:** Page Number

**Function:** Bits 1-3 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.

**Bit: 0        Name:** Enable Relocation and Protection

**Function:** 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<sub>8</sub>, 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  $777576_8$  and is illustrated in Figure 10-11.

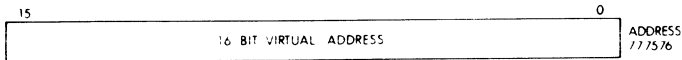


Figure 10-11 Format of Status Register 2 (SR2)

### 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 22-bit mapping, and SR3 bit 5 enables I/O mapping. The address of SR3 is  $772516_8$ . The format of status register 3 is shown below in Figure 10-12.

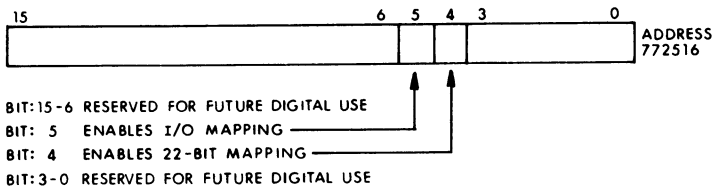


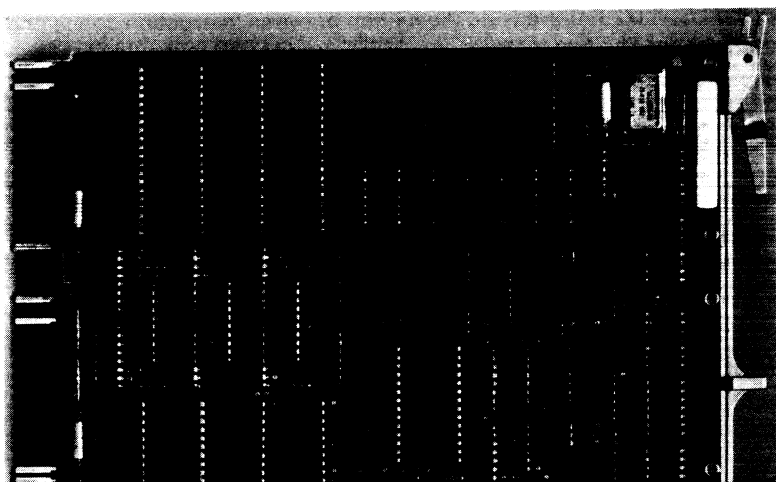
Figure 10-12 Format of Status Register 3 (SR3)

## MEMORY MANAGEMENT INSTRUCTIONS

Memory management provides communication between two spaces, as determined by the current and previous modes of the processor status word (PSW). The following instructions are directly applicable to memory management.

<b>Mnemonic</b>	<b>Instruction</b>	<b>Op Code</b>
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 11

# FPF11 FLOATING POINT PROCESSOR

The FPF11 floating point processor is a hardware option designed to operate with the LSI-11/23, PDP-11/23, or the PDP-11/23-PLUS central processor units to execute all 46 arithmetic operations of the PDP-11 floating point instruction set. Its dedicated high-speed data path and unique internal clock speed the execution of the PDP-11 floating point instruction set, and enhance the performance of the LSI-11/23 in high computation-oriented environments. The FPF11 is contained in one quad-height module, the M8188.

The FPF11's 64-bit wide data path is designed to avoid the use of complex arithmetic coding routines that would be required if a 16-bit CPU were to operate on the 32-bit or 64-bit operands. Its internal clock, controlled by the FPF11 microcode, generates variable-length microcycles so that each microword can be executed in a minimum amount of time. The FPF11 does not depend on the Memory Management Unit (MMU) for its scratched pad registers. It complements the KEF11-AA microcoded (chip) implementation with up to six times the improvement in floating point performance over the KEF11-AA.

Featuring both single- and double-precision (32- or 64-bit) capability, the FPF11 uses the same addressing modes and memory management (when present) as the CPU. Floating point processor instructions can reference the floating point accumulators, the CPU's general registers, or any location in memory.

For a complete list of the 46 floating point instructions implemented by the floating point processor, refer to Chapter 5.

### FEATURES — BENEFITS

- Performance increased five to six times that of KEF11 without FPF11 — satisfies applications that require high computational speed.
- FP11 instruction set — current KEF11 software is completely migratable to FPF11.
- KEF11 interface socket — no special hardware needed to add FPF11 to a LSI-11/23 system.

### SPECIFICATIONS

Identification	M8188
Type	Quad-size
Dimensions	26.5 cm × 28.8 cm × 1.27 cm (10.5 in × 8.9 in × 0.5 in)

Bus Loads	None
Operating Temperature:	5° C to 50° C (41° to 122°F)
Relative Humidity	10% to 95%, (no condensation)
Power Consumption	+5.0 Vdc, 7.5 A

## CONFIGURATION

The following is a general installation procedure for adding an FPF11 to a LSI-11/23 processor system. The FPF11 is installed in the backplane slot as illustrated in Figures 11-1 and 11-2. The M8188 (quad-height module) becomes an integral part of the CPU when it is installed in the backplane slot adjacent to the LSI-11/23 CPU. The FPF11 module connects to the CPU by a ribbon cable that plugs into the socket normally designated for the optional KEF11-AA floating-point processor chip. The FPF11 receives only power, not signals, from the backplane. The module operates from a single +5.0V DC source. The FPF11 also receives +12V over the ribbon cable that plugs into the floating point chip socket on the processor. This option is independent of the type of bus used by the processor. The FPF11 does not connect to the system bus, and has no effect on bus loading.

Before installing the FPF11, run system diagnostics to verify that the system receiving the option is working properly, then follow the steps listed below:

1. Turn the power off and reconfigure the system. Refer to Figures 11-1 and 11-2.

## WARNING

**To prevent damage to components, use the special handling procedures for MOS devices when performing the following steps.**

2. If present, remove the floating point processor chip from the CPU module. Refer to Figure 11-1 and 11-2 for its location.
3. To ensure proper bus grant continuity, configure the jumpers as indicated in Table 11-1. Refer to Figure 11-3 for the locations of the jumpers on the FPF11.
4. Insert the Berg connector on the ribbon cable into J1 of the FPF11 module.
5. Install the M8188 module in the vacant slot. For the LSI-11/23 system, this is slot 2, adjacent to the CPU (see Figure 11-1).
6. Fold the ribbon cable as shown in Figure 11-2.

7. Insert the 40-pin DIP plug of the ribbon cable into the floating point socket on the CPU. Note the position of pin 1 in Figure 11-2.

**NOTE**

Check for possible power supply overload before restoring power. The FPF11 module typically draws 7.5 A at 5 V dc.

8. Turn the power on and run the FPF11 diagnostics to verify the proper operation. Refer to Diagnostics section in this chapter for diagnostic information.
9. Run DEC-X11 to verify that the entire system (including FPF11) is operating properly.

	SLOT A	SLOT B	SLOT C	SLOT D
ROW 1	CPU			
ROW 2	FPF 11		M8188	
ROW 3	OPTION 3		OPTION 4	
ROW 4	OPTION 6		OPTION 5	

VIEW IS FROM MODULE SIDE OF CONNECTORS

Figure 11-1 FPF11 Module in LSI-11/23 System

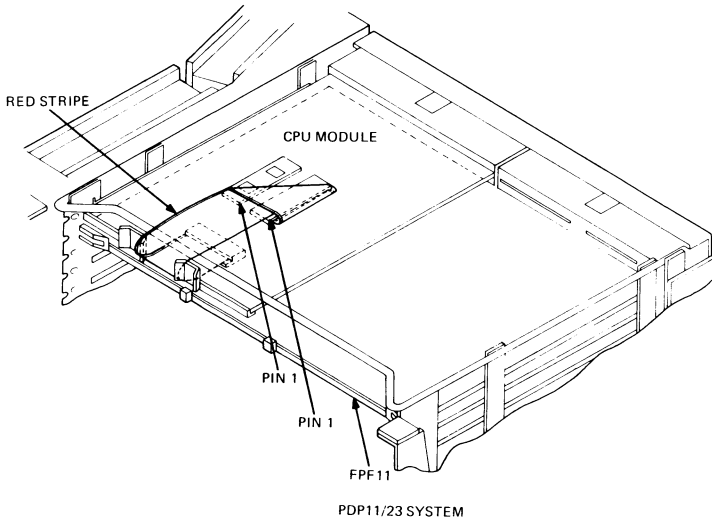


Figure 11-2 FPF11 Cable Layout in LSI-11/23 System

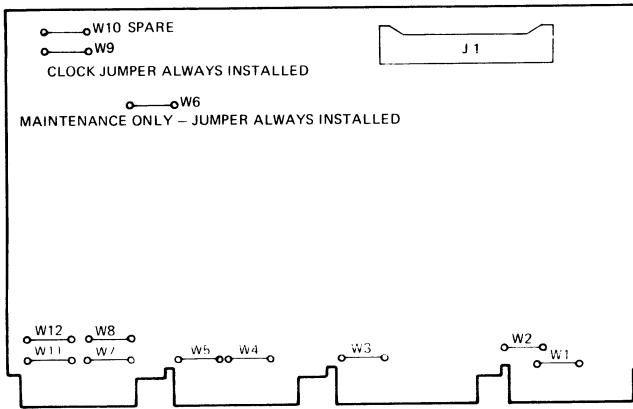


Figure 11-3 FPF11 Jumper Locations

Table 11-1 FPF11 Jumper Configurations

	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
Unibus	R	R	I	R	R	I	I	I	I	R	I	I
Q-Bus	I	I	R	I	I	I	R	R	I	I	R	R

NOTE: R = Jumper removed, I = Jumper installed.

### FPF11 Diagnostics

Two diagnostics are available to validate and diagnose the FPF11 option. The CPU tests should be run prior to running floating point diagnostics if there is any doubt about the CPU. Successful running of CPU tests does not rule out the possibility that a failure may cause only floating point instructions to fail. The two FPF11 diagnostics are listed below. These diagnostics must be run in the order listed because each test requires that the one preceding it was faultless. Otherwise, you may not identify correctly a failed microstep and location of its cause.



**MAINDEC CJFPAA (FPF11, No. 1)**

This diagnostic tests the following floating point instructions:

- LDFPS
- STFPS
- CFCC
- SETF, SETD, SETI, and SETL
- STST
- LDF and LDD (all source modes)
- STD (mode 0 and 1)
- ADDF, ADDD, and SUBD (most conditions)
- CMPD and CMPF
- DIVD and DIVF
- MULD and MULF
- MODD and MODF

**MAINDEC CFFPBA (FPF11, No. 2)**

This diagnostic tests the following floating point instructions:

- STF and STD (all modes)
- STCFD and STCDF
- CLRD and CLRF
- NEGF and NEGD
- ABSF and ABSD
- TSTF and TSTD
- NEGF, ABSF, and TSTF (all source modes)
- LDFBS (all source modes)
- LDCIF, LDCLF, LDCID, and LDCLD
- LDEXP
- STFPS (all destination modes)
- STCFL, STCFI, STCDL, and STCDI
- STEXP
- STST

**DESCRIPTION**

This description consists of two parts. The first part describes the interface between the CPU and the FPF11. The second part describes the internal operation of the FPF11.

**CPU Interface**

The CPU contains two internal busses, the Microinstruction Bus (MIB) and the Data/Address Lines (DALs) as illustrated in Figure 11-4. The MIB <15:00> is the control bus. It is used to carry the 16-bit CPU microinstructions. The DAL <15:00> is the data bus, which is used to carry data, addresses, and PDP-11 instructions. The FPF11 is connected to these busses by a ribbon cable.

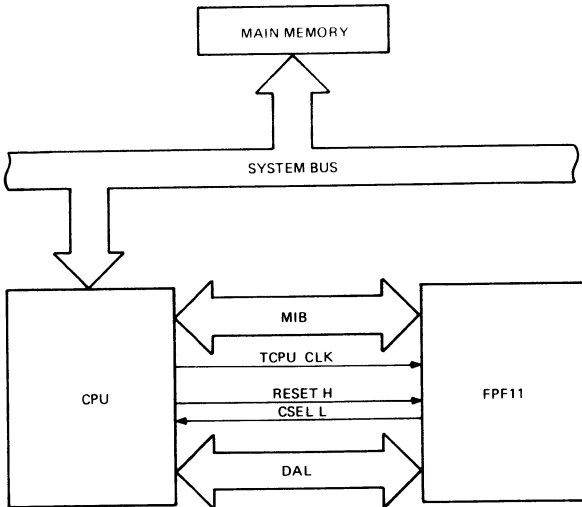


Figure 11-4 FPF11/CPU Interface

The CPU executes PDP-11 code by fetching an instruction and its operands from memory, performing the operation, and storing the result. It accomplishes this by executing CPU microinstructions. The FPF11 monitors the CPU microinstruction flow at its MIB interface. When the FPF11 decodes the microinstructions to be an instruction fetch (IFETCH), it knows the next piece of information on the DAL is a PDP-11 instruction, and it inputs this instruction into its floating point register (FPIR) in parallel with the CPU. The CPU uses the next two microcycles to examine the instruction. During this interval, the FPF11 sets up to process the instruction in case it turns out to be in the floating point class (i.e. it has an op code of 17XXXX).

At the end of the microcycles, the CPU microinstruction on the MIB tells the FPF11 whether or not it is in the floating point class. If it is not,

the FPF11 goes back to monitoring the CPU and awaits the next IF-ETCH. If it is a floating point instruction, the CPU passes control to the FPF11, which then becomes the microinstruction source on the MIB. At this point, the FPF11 has control over the CPU.

The FPF11 issues CPU microinstructions to move operands between main memory and the FPF11 as dictated by the PDP-11 floating point instruction on the FPIR. The operands are passed on the DAL data bus.

During the times when no CPU actions are required, the CPU receives a no operation (NOP) microinstruction. This occurs while the FPF11 operates on floating point data. The FPF11 may run at its own clock rate while the CPU receives NOPs at the CPU's clock rate. Resynchronization occurs prior to the return of control to the CPU, that is, upon successful completion of a PDP-11 floating point instruction. Control is passed back to the CPU by entering the CPU's service routine. This causes the FPF11 to return to its monitoring of CPU microinstructions. If the completion of a floating point instruction is unsuccessful, control is passed back to the CPU by entering the CPU's TRAP HANDLER microroutine. The trap vector associated with the floating point processor errors (244<sub>8</sub>) is sent to the CPU on the MIB. The FPF11 then returns to monitoring CPU microinstructions. The CPU handles the trap as directed by the interrupt service routine at 244<sub>8</sub>. The FPF11 stores the error code and address associated with the failing PDP-11 floating point instruction in its floating error code (FEC) and floating error address (FEA) registers, respectively. Software may use this information to recover from the error.

### Internal Operation

Figure 11-5 is a functional block diagram of the FPF11. Similar to the CPU, the FPF11 has both a control path and a data path. The control store outputs from a control path that is 104 bits wide.

Data, addresses, and PDP-11 floating point instructions are passed on the TBus <15:00>, which is the FPF11 internal bus.

Control functions are performed by the sequencer, the control store, the MIB interface, and the interface control logic. Data handling functions are performed by the DAL interface, TBus resources, and the microprocessor data path.

**Microcontroller: Sequencer and Control Store** — The sequencer and the control store together form the FPF11 microcontroller. The microcontroller directs all FPF11 activity by conditionally sequencing through the microcode in the control store.

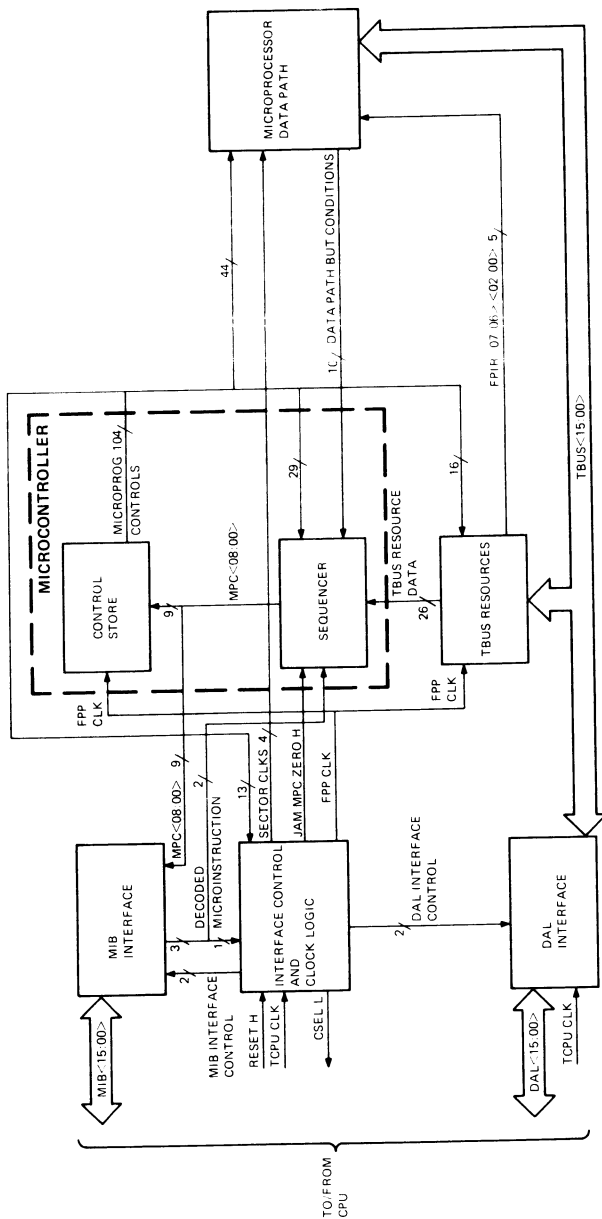


Figure 11-5 FFP11 Floating Point Processor Functional Block Diagram

The sequencer makes the decisions that control microprogram execution. It consists of branch and IR decode programmable logic arrays (PLAs) and the microprogram counter (MPC) logic. Information received from the TBus resources, MIB interface, microprocessor data path, interface control and clock logic, and bits of the previous control word cause the sequencer to output microaddresses on MPC <08:00>. These microaddresses are sent to the control store to select microinstructions for FPF11 control and to the MIB interface to select CPU microinstructions.

The control store logic is comprised of 13 512 x 8-bit ROMs which make up the control word. In addition, two more ROMs are used to hold CPU microinstructions. Hence, the microcontroller directs the execution of the current PDP-11 floating point instruction.

**MIB Interface** — The FPF11 uses the MIB interface to either monitor or control the CPU. While the CPU is executing PDP-11 instructions, this logic monitors the MIB, waiting to decode the CPU microinstruction associated with IFETCH. This decoded microinstruction is sent to the sequencer. This logic also decodes the CPU microinstruction, which indicates the start of a PDP-11 floating point instruction. While the FPF11 is executing PDP-11 floating point instructions, the microinstruction ROMs located here are addressed by MPC <08:00>, from the sequencer. This causes the microinstruction ROMs to output CPU microinstructions on the MIB <15:00>, and makes the CPU available to the FPF11 for use in accessing main memory. The direction of information flow out of the MIB interface is controlled by the microcontroller, and timed by the interface and control logic.

**Interface and Control Logic** — The interface control and clock logic is responsible for maintaining proper timing between the FPF11 and the CPU, for both data (DAL) and control (MIB). This logic provides the CPU with the signal that allows the FPF11 to be the CPU microinstruction source on the CSEL line (MIB), and also receives the RESET line from the CPU. In addition, internal FPF11 processing is timed here.

The interface and control logic receives decoded microinstructions from the MIB. When these instructions indicate a floating point instruction has been received by the CPU, the signal CSEL is sent to the CPU to inform it that the FPF11 is taking control. This logic times control and data signals entering or leaving the FPF11 at the MIB interface or DAL interface. When the CPU issues a RESET, or when the test for a floating point instruction decode from the CPU fails, the FPF11 microcode is reset to the CPU monitoring sequence. This is accomplished by the signal JAM MPC ZERO H from this logic to the sequencer.

The logic clock controls the speed at which the FPF11 runs. Normally, the FPF11 runs at the speed of the CPU. However, during certain arithmetic operations, the clock logic generates a faster clock for timing FPF11 operations, while the MIB interface runs at the speed of the CPU. The clock is controlled by the microcontroller, and is used to run each FPF11 microcycle at its fastest possible rate. The clock logic also synchronizes the FPF11 with the CPU prior to returning control to it.

**DAL Interface** — The DAL interface buffers the information received from the CPU's DAL and transfers the information from the TBus onto the DAL for use by the COU or storage in main memory. Information returned to the CPU consists of processed operands and associated addresses and error codes. The direction of the information flow is controlled by the microcontroller, and timed by the interface control and clock logic.

**TBus Resources** — The TBus resources supply the sequencer and the microprocessor data path with information necessary for data and control processing. The TBus resources are five registers, a counter, and a pair of constant ROMs. TBus resource data is passed to and from the TBus under the direction of the microcontroller.

The counter is used as a loop counter for microcode iterations, such as the multiply microroutine. Data is loaded into the counter from the TBus. The sequencer uses an output of the counter, CNTR BORROW L, to control microprogram flow. The counter outputs are also available on the TBus for use in the microprocessor data path.

The two constant ROMs contain fixed-value numbers required for certain floating point operations.

The five TBus registers are as follows:

- |       |  |
|-------|--|
| FPIR  | The floating point instruction register (FPIR) contains the op code, accumulator, source/destination, and floating source/floating destination information.  |
| FD/FL | This register specifies whether the mode is single- or double-precision floating point format (FD), or single- or double-precision integer format (FL).  |
| FPS   | The floating point status (FPS) register consists of two segments, FPS1 and FPS2. The microcontroller tests the bits in FPS1 in order to control microprogram flow. Floating point condition codes are loaded into the 4-bit FPS2. These codes reflect the condition |

of the last arithmetic operation (that affected condition codes) performed in the microprocessor data path.

SFR

The status flags register (SFR) is where the microcode stores special microcode conditions as status bits. The microcontroller uses these bits to control microprogram flow. They allow a limited amount of microcode sharing to occur, similar to subroutines.

**Microprocessor Data Path Logic** — This logic performs the floating point arithmetic and data manipulations. The logic consists of 16 AM2901 bipolar microprocessors and support circuitry. The support circuitry consists of fast carry logic, shift linkage logic, and address multiplexers for scratch-pad registers.

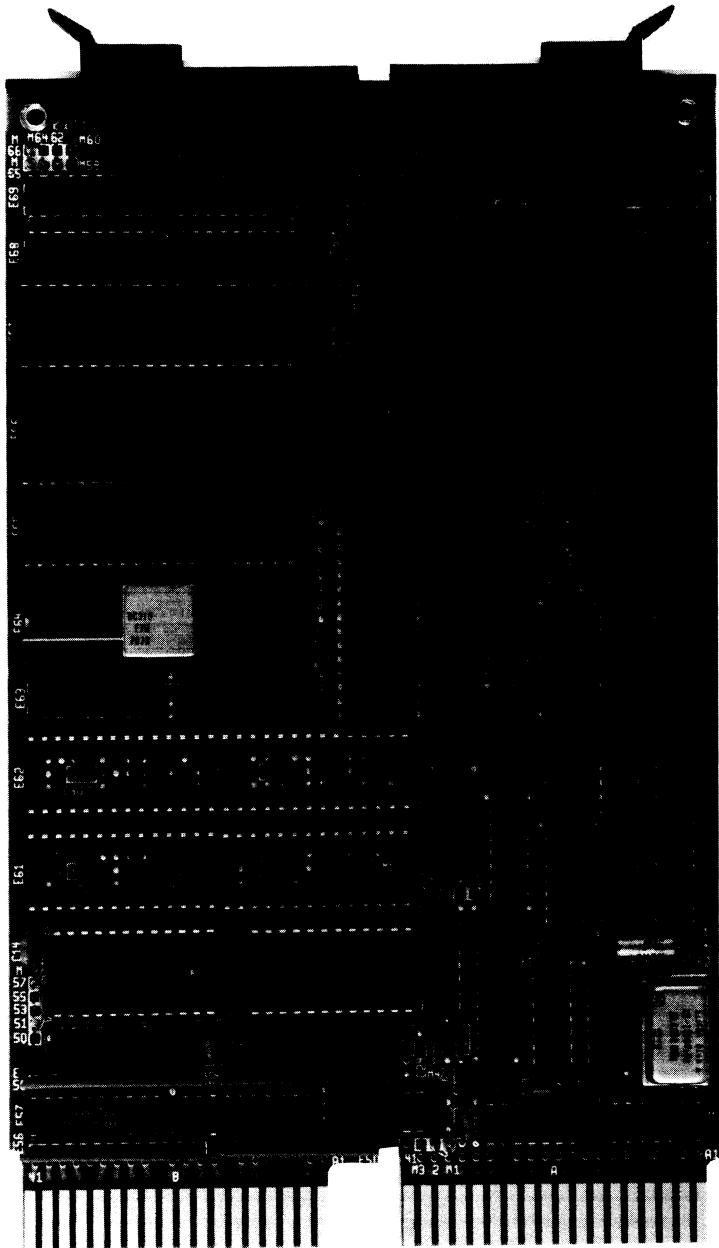
Operands are moved to and from an AM2901 bipolar microprocessor in 16-bit words by way of the TBus. The microcontroller directs the placement of the operands at their specified locations within a scratch-pad register. Once the operands are loaded, they are manipulated by the microcontroller to arrive at resultant operands. These operands are then stored in the scratch-pad register until they are moved onto the TBus.





**section II**  
**board-level**  
**components**





## CHAPTER 12

# FALCON SBC-11/21 MICROCOMPUTER

### INTRODUCTION

The FALCON SBC-11/21, KXT11-AA, is DIGITAL's first single-board microcomputer. Designed for use in low-cost, dedicated, PROM-based control applications, the FALCON executes software out of ROM, PROM, or EPROM memory. The stand-alone FALCON offers the DIGITAL-designed 16-bit Micro/T-11 Microprocessor Unit (MPU), 4 KB of RAM, up to 32 KB of ROM, two asynchronous serial line units with programmable baud rates, and 24 lines of programmable parallel I/O-- all of which are packaged on one dual-height board module.

While retaining full PDP-11 hardware and software compatibility, the very compact SBC-11/21 extends the low-end of DIGITAL's PDP-11 family of computers. The Micro/T-11 MPU executes the PDP-11 base-level instruction set, (minus the MARK instruction) which consists of over 400 instructions and 12 powerful addressing modes.

FALCON microcomputer design has been optimized to support the execution of ROM, PROM, or EPROM-based application software. The MicroPower/Pascal runtime operating system is the only operating system software that supports the FALCON. MicroPower/Pascal is comprised of a new language compiler, a modular operating system for the target application, and powerful debugging tools for use in the program development environment. Program development for MicroPower/Pascal is implemented under RT-11.

### FEATURES — BENEFITS

- 0-64 KB of address space — allows flexibility of expanding to LSI-11 Bus memory if necessary.
- Compact size — allows versatile packaging.
- Stack processing — simplifies handling of structured data, subroutines, and interrupts.
- PDP-11 instruction set — provides power and flexibility of world's most popular computer architecture.
- Two asynchronous serial I/O ports — allows downline program loading, simultaneous debugging and monitoring of programs, as well as communication with terminal.

- Two bidirectional parallel ports — provide high-speed parallel interface to external logic.
- LSI-11 Bus interface — allows communication with all LSI-11 Bus compatible products.
- Four memory maps — allow user to design memory application requirements.
- Memory battery backup hooks — allow user to protect data in RAM memory from power outages.
- Real-time clock — can precisely synchronize real-time events with on-board clock.
- Four interrupt levels — allow user to assign priorities of device interrupts; simplify exception handling.
- 4 KB of static RAM (standard) — provides application program scratch pad.
- 4 28-pin memory sockets — support up to 32 KB of EPROM with 4 KB RAM memory; or 16 KB of EPROM with 8 KB of RAM.
- On-board LED — provides program-controlled visual indicator.
- Power-up self-test diagnostics — provides assurance of proper module operation.
- On-board charge pump for negative voltage — provides a  $-12\text{ V}$  for EIA interface.
- Wake-up circuit — allows operation with off-the-shelf (non-DIGITAL) power supplies. Gives a valid initialization at power-up.
- Programmable baud rate — dynamic baud rate selection under program control.

## SPECIFICATIONS

### Power Requirements

Operational Power (measured at module fingerspins BV1 and BT1)	+5 V $\pm 5\%$ ; 2.8 A max. +12 V $\pm 5\%$ ; 0.1 A max. (or, 1.1 A max. with DLV11-KA options installed)
Battery Backup	+5 VB $\pm 5\%$ ; 260 mA max.
Bus Loads	AC 2.4 unit loads DC .5 unit loads

### Physical Characteristics

Height	13.2 cm (5.19 in)
Width	1.27 cm (0.5 in)
Length	22.7 cm (8.94 in)
Weight	.34 kg (12 oz) max.
Size	Double-height

### Operating Environment

Temperature	5°C to 60°C
Airflow	Adequate airflow must be provided to limit the temperature rise across the module to: <5°C for 60°C inlet temperature <10°C for inlet temperature below 55°C
Relative Humidity	10%–90% noncondensing
Altitude	up to 26.2 km note: Derate the maximum operating temperature by 1.8°C for each 1000 m of altitude above sea level

### Storage Environment

Temperature	–40°C to 65°C
Relative Humidity	10%–95% noncondensing
Altitude	up to 98.4 km

**CONFIGURATION**

The FALCON SBC-11/21 microcomputer can reside either in a back-plane or operate as a stand-alone module, depending on the specific application and requirement of its user.

The FALCON SBC-11/21 microprocessor module contains wire-wrap pins which must be configured to meet the requirements of specific modes of operation. Jumper wires connecting the wire-wrap pins can either be removed or installed, depending on the configuration needed. The numerical identifications and locations of the wire-wrap pins are illustrated in Figure 12-1. The wire-wrap pins and their functions are listed in Table 12-1 in order of the features they support. The configured features discussed in this chapter will include Power-up, starting address, a few examples of parallel I/O buffering and memory maps, and serial line unit specifications. For information on configuring battery backup, interrupts, and all types and sizes of parallel I/O and memory examples, please consult the FALCON SBC-11/21 User Guide.

**Table 12-1 Configuration Pin Definitions**

<b>Pin</b>	<b>Function</b>	<b>Description</b>
	Nonmaskable Interrupt and Trap to the Restart Address	
M5		+5 Vdc voltage level
M6		-CTMER interrupt request input (edge sensitive)
M7		Timeout error (TMER) output
M8		-CTMER interrupt enable/disable
M9		Interrupt Acknowledge (-IAK) output (active low)
M10		System GND
M11		High logic level (+3 Vdc)
	Serial Line Unit #1	
M12		System GND
M13		Transmit side of BHALT line transceiver

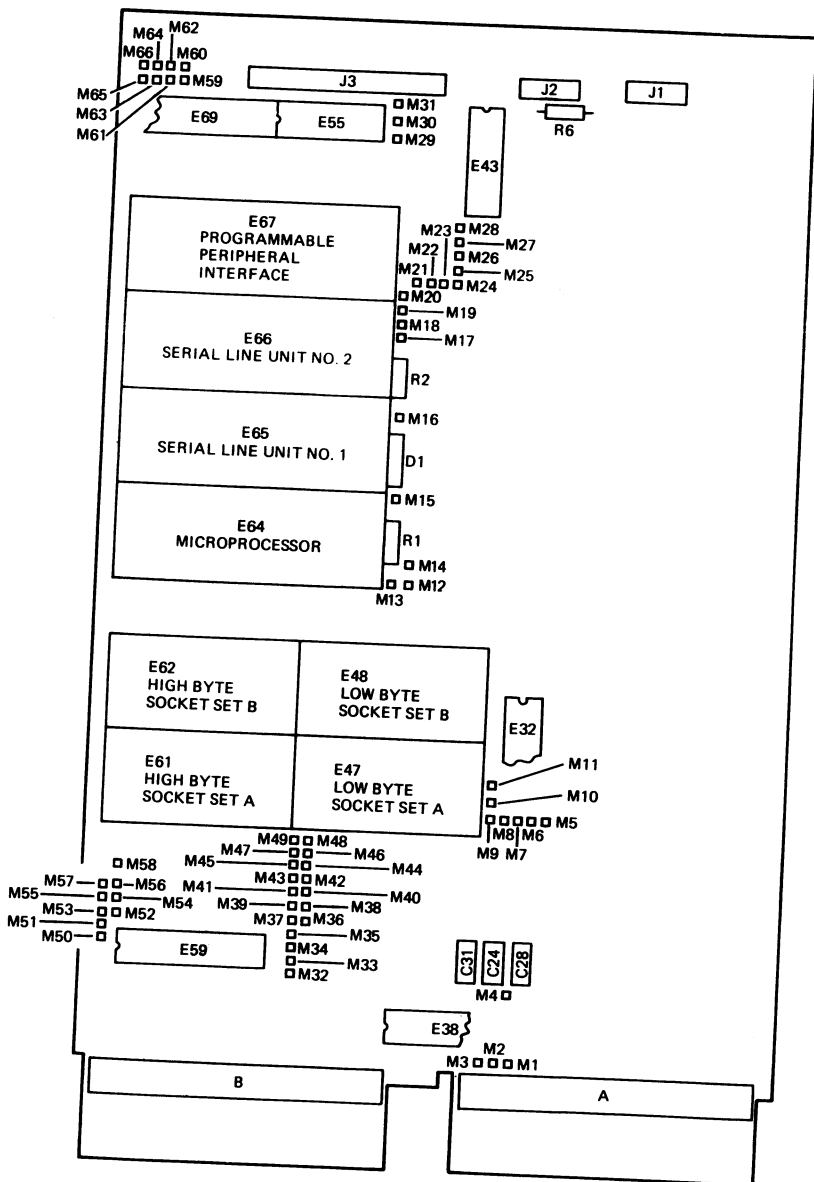


Figure 12-1 FALCON SBC-11/21 Module Map

Pin	Function	Description
M14		Serial Line unit #1 Break Detect Interrupt request output
	Power-Up	
M15		System +5 V power (wake up circuit diode)
M16		Wake up circuit diode disable (anode side)
M17		Transmit side of BEVNT line transceiver
M18		50 Hz real-time clock output
M19		60 Hz real-time clock output
M20		800 Hz real-time clock output
	Memory map Decoder	
M21		High logic level (+3 Vdc)
M22		Memory map select (LSB)
M23		Memory map select (MSB)
M24		System GND
	Start Address (Mode Register)	
M25		Mode register bit 13
M26		Mode register bit 14
M27		Mode register bit 15
M28		High logic level (+3 Vdc)
	Memory	
M32		Address line 11
M33		High logic level for Vpp input
M34		Socket set A high byte pin 23
M35		Socket set B high and low byte pin 27
M36		Socket set A high and low byte pin 20
M37		Socket set B high byte pin 23
M38		Socket set B high and low byte pin 20
M39		Socket set B high byte pin 22
M40		Socket set B low byte pin 22



Pin	Function	Description
M41		Socket set A low byte pin 22
M42		Socket set B low byte pin 23
M43		Socket set A low byte pin 23
M44		Socket set A high and low byte pin 2
M45		Socket Set B high and low byte pin 2
M46		Address line 13
M47		Socket Set B Chip Select (-CSKTB)
M48		Socket Set A high and low byte pin 27
M49		Address line 12
M50		System GND
M51		Read Strobe (-Read)
M52		Write low byte strobe (-WLB)
M53		Static RAM high byte Output Enable (OE)
M54		Static RAM low byte Output Enable (OE)
M55		High byte write strobe (-WHB)
M56		Socket set A Chip Select (-CSKTA)
M57		Socket set A high byte pin 22
M58		Socket set A and B high and low byte pin 21
	Parallel Input/Output	
M59		Port B Buffer direction control
M60		System GND
M61		Port C buffer output from J3 pin 5 input
M62		Port C buffer output from J3 pin 7 input
M63		Port C PC4 output (8255A-5 pin 13)
M64		Port C PC6 output (8255A-5 pin 11)
M65		High logic level (+3 Vdc)
M66		Port A buffer direction control

The standard factory configurations are described in Table 12-2. Jumpers are installed only on designated pins. No wire-wrap jumpers are installed to the pins listed under no connections.

**Table 12-2 Standard Factory Configurations**

<b>Function</b>	<b>Install Jumpers Between</b>
Standard LSI-11 Bus Power (No Battery Backup)	M1 and M2 M1 and M4
Power-Up Circuit Enabled	No jumpers
Start Address (Mode Register)	M25 and M26
Start address 1000	M26 and M24
Restart address 10004	M27 and M28
Memories:	M22 and M23
Memory Map 1	M23 and M24
2K X 8 INTEL EPROM	M33 and M43 M33 and M34 M41 and M57 M56 and M36 M51 and M57 M32 and M58 M33 and M42 M33 and M37 M41 and M39 M47 and M38 M51 and M40
Interrupts:	
Time out traps to restart address except during LSI-11 Bus IAK.	M9 and M8 M7 and M6
SLU#1 Break asserts BHALT and BHALT is received as level 7 interrupt (vector 140)	M13 and M14 M30 and M31 M19 and M17
SLU#2 60 Hz Real-Time Clock asserts LSI-11 BEVNT	
Parallel I/O	M59 and M60
Port A Receive data	M65 and M66
Port B Transmit data PC4 input	M61 and M63
No Connection to pins M3, M5, M10, M11, M12, M15, M16, M18, M20, M21, M35, M44, M45, M46, M48,	

M49, M50, M52, M53, M54, M55,  
M62, M64

### Power-Up

The FALCON SBC-11/21 contains a wake up circuit to provide a power-up sequencing routine whenever power is applied. When using the KXT11-AA in a DIGITAL-supplied box (except the BA11-VA), the KXT11-AA wake up circuit must be disabled. To do this, a jumper wire must be installed between M15 and M16. This jumper wire should be removed when power supplies which do not provide power-up sequencing are used. The KXT11-AA module requires that the +5 Vdc and +12 Vdc power supplies have a rise time of less than 50 ms. DIGITAL boxes provide both power-up and power-down sequencing signals. If battery backup is desired, a power supply with both power-up and power-down sequencing signals is required, or a KPV11 can be used.

### Starting Address

The FALCON SBC-11/21 start and restart addresses are selected by the user via the mode register. After the FALCON SBC-11/21 has been powered up, the mode register loads the start address into the microprocessor. The microprocessor then loads this value into R7, the program counter, as the first fetch address. Wire-wrap pins M24, M25, M26, M27, and M28 are used to configure the mode register. These wire-wrap pin locations are shown in Figure 12-1. The user can then select any one of eight available start addresses. Table 12-3, the Mode Register Configuration table, lists the available addresses and jumper connections required for each address. The restart address is always the start address incremented by four.

**Table 12-3 Mode Register Configurations**

Start Address	Restart Address	Connect M27 to	Connect M26 to	Connect M25 to
000000	000004	M28	M24	M28
010000	010004	M28	M24	M24
020000	020004	M24	M28	M28
040000	040004	M24	M28	M24
100000	100004	M24	M24	M28
140000	140004	M24	M24	M24
172000	172004	M28	M28	M28
173000	173004	M28	M28	M24

**Parallel I/O**

The parallel I/O is controlled by the programmable peripheral Interface (PPI) and connects to the user's interface through the J3 connector. For locations and functions of these wire-wrap pins, refer to Table 12-1. The factory configuration jumpers required for configuration are M59, M60, M61, M63, M65, and M66. The direction of port A and port B transceivers are dependent upon the logic level connected to M59 and M66. Wire-wrap pin 66 connects to port A through a 200 ns minimum rising edge time delay circuit. When M65 (+3 Vdc) is jumpered to pins M59 and M66, port A and port B buffers are input to the PPI from the J3 connectors. When M60 (Ground) is jumpered to pins M59 and M66, port A and port B buffers are output from the PPI to the J3 connector. The direction of port A and port B can also be controlled by the user program.

The user can control the direction of the transceivers. Wire-wrap pins M63 and M64, when not jumpered to M61 or M62, can be jumpered to M59 and M66 to allow the user's program to control the direction of the transceivers via PPI port C outputs. In cases when not using wire-wrap pins M63 and M61 or M64 and M62 to control the direction of ports A and B, then a jumper connected between M63 and M61, plus a jumper connected between M64 and M62 allows port C lines to be used as inputs to the PPI from the J3 connector.

**NOTE**

If pins M61, M62, M63, or M64 are used for program control of ports A and B, then ensure that the PPI and the buffer do not contend as driver output to driver output.

The Programmable Peripheral Interface (PPI) functions in combinations of either Mode 0, Mode 1, or Mode 2. The jumper configurations and the handshake signals for each of these modes are listed in Tables 12-4, 12-5, and 12-6. For more detailed information for the Programmable Peripheral Interface refer to the KXT11-AA User Guide.

**Table 12-4 Mode 0 Buffer Configuration and Handshake**

PPI I/O	Mode 0 Configuration		Port C Controlled
	Input Status	Output Status	
Port A	M66 to M65	M66 to M60	M66 to M64 or M63

<b>PPI I/O</b>	<b>Input Status</b>	<b>Output Status</b>	<b>Port C Controlled</b>
Port B	M59 to M65	M59 to M60	M59 to M64 or M63
PC7	N/A	Output	
PC6	M64 to M62	M64NC, M62NC	M62NC
PC5	N/A	Output	
PC4	M62 to M61	M63NC, M61NC	M61NC
PC3	N/A	Interrupt A	
PC2	Input	N/A	
PC1	N/A	Output	
PC0	N/A	Interrupt B	

**Table 12-5 Mode 1 Buffer Configuration and Handshake**

<b>Mode 1 Configuration and Handshake</b>			
<b>PPI I/O</b>	<b>Input Status</b>	<b>Output Status</b>	<b>Port C Controlled</b>
Port A	M66 to M65	M66 to M60	N/A
Port B	M59 to M65	M59 to M60	M59 to M64 or M63
PC7	Output	Output Buffer A Filled	
PC6	Input	Acknowledge A	M62NC
PC5	Input Buffer A Filled		Output
PC4	Strobe A	Output M63NC	M61NC
PC3	Interrupt A/Out	Interrupt A/Out	
PC2	Strobe B	Acknowledge B	

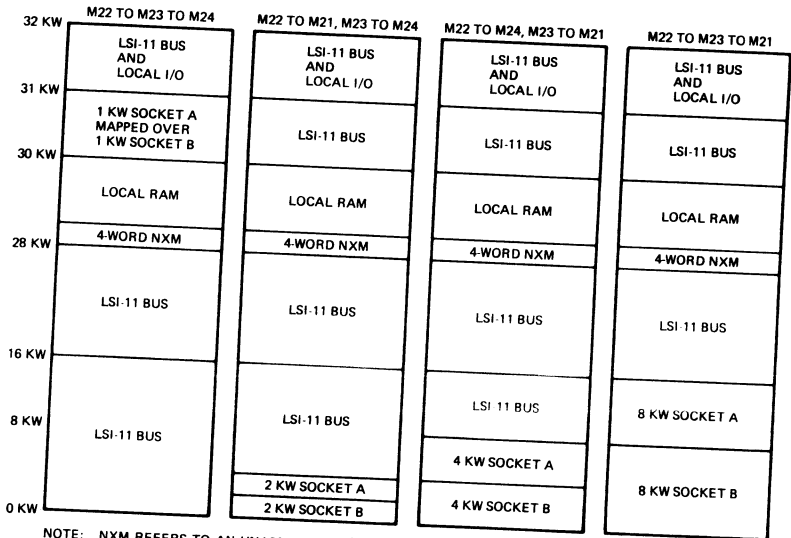
PPI I/O	Input Status	Output Status	Port C Controlled
PC1	Input Buffer B Filled	Output Buffer B Filled	
PC0	Interrupt B/Out	Interrupt B/Out	

**Table 12-6 Mode 2 Buffer Configuration and Handshake**

Mode 2 Configuration and Handshake		
PPI I/O	Input Status	Output Status
Port A	Bidirectional bus	M66 to M64 to M62
Port B	N/A	N/A
PC7	N/A	Output Buffer A filled
PC6	Acknowledge A	N/A
PC5	N/A	Input Buffer A filled
PC4	Strobe A	N/A
PC3	N/A	Interrupt A/Out
PC2	Input	N/A
PC1	N/A	N/A
PC0	N/A	N/A

### Memory Address

The memory system for the KXT11-AA module consists of the LSI-11 Bus, 4 KB of local RAM and four 28-pin sockets that will accept either 24-pin or 28-pin industry standard +5 V memory chips. These chips are provided by the user and can be EPROMs, PROMs, ROMs, or static RAM. The sockets will accept 1K × 8, 2K × 8, 4K × 8, and 8K × 8 PROMs/EPROM or 2K × 8 static RAM. The wire-wrap pins used to configure memory are described in Table 12-1. Four examples of configuring the memory address are illustrated in the Figure 12-2, below.



NOTE: NXM REFERS TO AN UNASSIGNED LSI-11 BUS ADDRESS.

Figure 12-2 Memory Address

Tables 12-7 and 12-8, below, show the jumper configurations for some of the common ROMs, PROMs, and EPROMs that can be used in socket A and socket B. If the ROM, PROM, or EPROM you are trying to configure is not listed here, refer to the KXT11-AA User Guide. This guide provides all the configurations for the sockets you wish to configure for your chips.

Table 12-7 Socket A Configurations for Read-Only Memories

Vendor	Device Size	Package Size	Jumper Connection							
			M43	M48	M44	M34	M57	M38	M41	M58
INTEL	1K x 8	24 PROM	M56	NC	NC	M56	M51	M33	M51	M33
INTEL	1K x 8	24 EPROM	M5	NC	NC	M5	M51	M56	M51	M50
INTEL	2K x 8	24 EPROM	M5	NC	NC	M5	M51	M56	M51	M32
INTEL	4K x 8	24 EPROM	M49	NC	NC	M49	M51	M56	M51	M32
INTEL	8K x 8	28 EPROM	M49	M33	M46	M49	M51	M56	M51	M32

Vendor	Device Size	Package Size	Jumper Connection							
			M43	M48	M44	M34	M57	M38	M41	M58
TI	1K x 8	24 EPROM	M5	NC	NC	M5	M51	M56	M51	M33
Signetics	1K x 8	24 PROM	M56	NC	NC	M56	M51	M33	M51	M33
TI	2K x 8	24 EPROM	M5	NC	NC	M5	M51	M56	M51	M32
TI	8K x 8	28 EPROM	M46	M50	M56	M46	M51	M49	M51	M32
MOSTEK	2K x 8	24 EPROM	M5	NC	NC	M5	M51	M56	M51	M32
MOSTEK	8K x 8	28 EPROM	M49	NC	M46	M49	M51	M56	M51	M32

**Table 12-8 Socket B Configurations for Read-Only Memories**

Vendor	Device Size	Package Size	Jumper Connection							
			M42	M35	M45	M37	M39	M38	M40	M58
INTEL	1K x 8	24 PROM	M47	NC	NC	M57	M51	M33	M51	M33
INTEL	1K x 8	24 EPROM	M5	NC	NC	M5	M51	M47	M51	M50
INTEL	2K x 8	24 EPROM	M5	NC	NC	M5	M51	M47	M51	M32
INTEL	2K x 8	24 EPROM	M32	NC	NC	M32	M51	M47	M51	M33
Signetics	2K x 8	24 EPROM	M32	NC	NC	M32	M51	M47	M51	M33
INTEL	4K x 8	24 EPROM	M49	NC	NC	M49	M51	M47	M51	M32
INTEL	8K x 8	28 EPROM	M49	M33	M46	M49	M51	M47	M51	M32
TI	1K x 8	24 EPROM	M5	NC	NC	M5	M51	M47	M51	M33
Signetics	1K x 8	24 EPROM	M47	NC	NC	M47	M51	M33	M51	M33
TI	2K x 8	24 EPROM	M5	NC	NC	M5	M51	M47	M51	M32



Vendor	Device Size	Package Size	Jumper Connection													
			M42	M35	M45	M37	M39	M38	M40	M58						
TI	8K x 8	28 EPROM	M46	M50	M47	M46	M51	M49	M51	M32						
MOSTEK	2K x 8	24 EPROM	M5	NC	NC	M5	M51	M47	M51	M32						
MOSTEK	8K x 8	28 EPROM	M49	NC	M46	M49	M51	M47	M51	M32						

The following Tables 12-9 and 12-10 are partial lists of static Random Access Read/Write Memories (RAM) which can be configured into the KXT11-AA memory sockets A and B.

**Table 12-9 Socket A Configurations for RAM Memories**

Vendor	Device Size	Package Size	Jumper Connection							
			M43	M48	M44	M34	M57	M36	M41	M58
MOSTEK	2K x 8	24	M52	NC	NC	M55	M53	M56	M54	M32
Hitachi	2K x 8	24	M52	NC	NC	M55	M53	M56	M54	M32

**Table 12-10 Socket B Configurations for RAM Memories**

Vendor	Device Size	Package Size	Jumper Connection							
			M43	M48	M44	M34	M57	M36	M41	M58
MOSTEK	2K x 8	24	M52	NC	NC	M55	M53	M47	M54	M32
Hitachi	2K x 8	24	M52	NC	NC	M55	M53	M47	M54	M32

### Serial Line Addresses and Parallel Port Addresses

The parallel port and serial line unit addresses are determined by a Field Programmable Logic Array (FPLA), which is the decoding device for applied address bits, and cannot be altered. The FPLA allows a choice of four starting addresses for the four 28-pin sockets. The vectors and priorities for all on-board interrupts are determined by a PROM that cannot be altered. The interrupt priority for all LSI-11 Bus devices is subject to the daisy-chain priority, however the vector is read in from the interrupt device. Tables 12-11 and 12-12 below list the SLU and parallel port addresses and device interrupt vector and priorities.

**Table 12-11 SLU and Parallel Port Addresses**

SLU 1 transmitter	177560
SLU 1 transmit buffer	177562
SLU 1 receiver	177564
SLU 1 receive buffer	177566
SLU 2 transmitter	176540
SLU 2 transmit buffer	176542
SLU 2 receiver	176544
SLU 2 receive buffer	176546
Parallel port 1	176200
Parallel port 2	176202
Parallel port 3 or STATUS	176204
Parallel Control Word Register	176206

**Table 12-12 Device Interrupt Vector and Priorities**

Device	Priority	Vector
Power-fail		24
BHALT, SLU1 BREAK or Time out	7.1	140
BEVNT	6.1	100
SLU2 receiver	5.4	120
SLU2 transmitter	5.3	124
Parallel request B (PC0)	5.2	130
Parallel request A (PC3)	5.1	134
SLU1 receiver	4.3	60
SLU1 transmitter	4.2	64
LSI-11 Bus device	4.1	read from LSI-11 Bus

**DESCRIPTION**

The KXT11-AA contains a 16-bit microprocessor that controls the read/write, fetch, direct memory access, and interrupt transactions. The microprocessor operates as the central processing and control unit, and is contained within a 40-pin LSI chip. The module provides the necessary control circuits to interface with the LSI-11 Bus. The microprocessor functions as a bus master, a bus slave, a bus arbitrator, and allows a DMA master to access the on-board functions. The KXT11-AA functional block diagram, Figure 12-3, below, provides an overview of the module functions and how they are related.



The FALCON SBC-11/21 microcomputer consists of a microprocessor connected to serial line units, RAM memory, ROM memory, and a parallel I/O interface via on-board TDAL bus. The KXT11-AA TDAL bus is a 16-bit internal multiplexed data and address bus which is common to all local memory and I/O, and to the LSI-11 Bus transceivers. The TDAL bus can access the LSI-11 Bus BDAL bus--the LSI-11 Bus (backplane) multiplexed data and address lines--by the bus control function illustrated by the dashed lines in Figure 12-3. An internal and external vectored interrupt structure will relinquish the bus for Direct Memory Access (DMA). For a more thoroughly detailed description of the functional theory of KXT11-AA microprocessor functional and operational inputs, signals, interrupts, transceivers, and logic, refer to the KXT11-AA User Guide.

The FALCON's microprocessor contains eight 16-bit general-purpose registers (R0-R7). R6 operates as the stack pointer (SP), and the R7 operates as the microprocessor program counter (PC). The microprocessor can address up to 64 KB of physical memory. A special purpose status register contains the current Processor Status Word (PSW). The operating characteristics of the microprocessor are predetermined by the mode register. The user can select a start and restart address from eight possible selections by setting bits 13, 14, and 15 of the mode register.

### **Serial Line Interface Units**

The FALCON SBC-11/21 contains two asynchronous Serial Line Units (SLU) to provide serial I/O interface. The serial I/O interface allows the microprocessor memory to be loaded with data either down-line from a host or locally from serial mass storage devices such as DIGITAL's TU58 tape cartridge system. The ports also provide direct communication with terminals, lineprinters, and other asynchronous serial devices. These ports are DL-11 software-compatible and have eight programmable line speeds from 300 to 38.4K bits per second. The ports are compatible with EIA RS-423 and RS-232C electrical standards. For local communications, each port supports DIGITAL's DLV11-KA 20 mA communications option.

The SLUs transmit or receive 8-bit, byte-oriented data with no parity and only one stop bit. The transmitter and receiver must operate at the same baud rate.

SLU 1 provides the XDL1 and RDL2 interrupts for transmit and receive, and the BREAK output which is wired to pin M14. The user can jumper the BREAK output to the HALT interrupt (pin M13), and use SLU 1 as a system console.

SLU 2 provides the XDL2 and RDL2 interrupts for transmit and receive, and three real-time clock interrupts at 50 Hz, 60 Hz, and 800 Hz. These interrupts are wired to pins M18, M19, and M20 for use with the BEVNT interrupt (pin M17).

### **Parallel I/O Interface**

In addition to two serial interface ports, the FALCON SBC-11/21 also provides a 24-line parallel I/O port. This port provides the data paths and control signals to transmit information between the microcomputer and control panels, data multiplexers, or custom logic integrated circuits.

The parallel port is divided into three 8-line ports. Ports A and B provide a 16 line data path. Port A can operate as either an 8-bit input or output port, or as an 8-bit bidirectional data bus. Port B functions as either an 8-bit input or output bus. Port C is dedicated to control signals, handshaking, and interrupt requests (there is one interrupt available for each data port). Port C also controls the onboard LED (light emitting diode), a programmable visual indicator that can be useful in debugging software, as a failure indicator or as a power-up diagnostic indicator.

The port provides excellent noise immunity and signal quality on the lines because it is buffered with P-N-P hysteresis input receivers and with high-current drivers. Connections to the port are made through a 30-pin shrouded header.

### **LSI-11 Bus Interface**

The FALCON SBC-11/21 provides extensive I/O expansion capabilities through the LSI-11 Bus interface. A/D (analog to digital) and D/A (digital to analog) converters, complex parallel I/O structures, modem-controlled communications, and data concentration lines may be easily configured through this interface. The LSI-11 Bus interface supports DIGITAL's LSI-11 16-bit multiplexed address/data bus and all of DIGITAL's LSI-11 Bus-compatible hardware. The bus interrupt priority is established by the physical position of each device on the bus. The device closest to the processor has the highest priority.

The LSI-11 Bus interface arbitrates DMA (direct memory access) requests to allow DMA transfers between devices on the bus, and between devices on the bus and the FALCON's onboard memory.

### **RAM Memory**

The FALCON SBC-11/21 includes 4 KB of static RAM organized as 2,048 16-bit words. This RAM consists of a 2K X 8-bit high byte chip and a 2K X 8-bit low byte chip. Battery backup hooks are included for

battery backup voltages which protect data in the onboard RAM from power outages.

### **ROM Memory Sockets**

Four 28-pin memory sockets provide the user with a choice of accepting either 28-pin or 24-pin industry-standard +5 V memory chips for read-only memory (ROM, PROM, or EPROM) and/or additional RAM memory. These sockets can accommodate up to 32 KB of ROM, and up to 8 KB of static RAM. The two socket sets designated A and B, and each has a high byte socket as well as a low byte socket.

### **NOTE**

If 1K X 8 PROMs are configured into a 4 KB memory and used in 8 KB, 16 KB, or 32 KB memory mapped configurations, addresses above the 4 KB boundary will wrap around into the 4 KB boundary.

When using 4K X 8 or 8K X 8 devices, the device pattern must be programmed into the device relative to address bits AD12 and AD13 when the device is addressed into a memory map. That is, the highest address portion of the program may have to reside in the lower physical half of the PROM.







## CHAPTER 13

# LSI-11/23 MICROCOMPUTER

### INTRODUCTION

The LSI-11/23 microcomputer (KDF11-AA) offers the power and performance of a mid-range minicomputer, but at a lower price. Capable of addressing up to four megabytes of main memory, this 16-bit, high-performance microcomputer, contained on one dual-height multilayer module (M8186), utilizes the latest MOS/LSI technology, and communications between the CPU, memory, and peripherals is passed along through DIGITAL's industry-standard LSI-11 Bus.

The LSI-11/23 also offers memory management, the FP-11 instruction set, and a new floating point accelerator module, the FPF11, as options.

The LSI-11/23 microcomputer is compatible with other LSI-11 microcomputers and software-compatible with the PDP-11 family. A wide range of software is available, including programming languages, diagnostic software, and operating systems. Table 13-1 illustrates the available LSI-11/23 configurations.

### FEATURES — BENEFITS

- No on-board memory — flexibility to match RAM/ROM size to requirements.
- Compact, double-height module size — allows for versatile packaging.
- ODT console emulator — ease of program debugging.
- Direct addressing of 2 million 16-bit words or 4 million 8-bit bytes — provides flexibility in defining data structures.
- 87 PDP-11 standard instructions — provide powerful and convenient programming.
- 8 addressing modes for specifying operands — allow for absolute, deferred, autoincrement, autodecrement, and index register references.
- 8 internal general-purpose registers for use as accumulators and for operand addressing — provide flexible programming techniques.
- Stack processing — creates convenient handling of structured data, sub-routines, and interrupts.
- Byte-oriented instructions — provide efficient processing of 8-bit characters without the need to rotate, swap, or mask.

- Vectored interrupts — provide fast interrupt response without device polling.
- Four-level interrupt bus structure — allows the priority of bus options for each level to be conveniently determined by their physical locations on the bus.
- Direct Memory Access (DMA) — allows peripherals to access memory without interrupting processor operation.
- Asynchronous bus operation — allows processor and system components (memory and peripherals) to run at their highest possible speeds.
- Memory parity errors are recognized during every data-in bus cycle — provides for overall system integrity.
- Power-fail and automatic restart hardware — detects and protects against ac power fluctuations.
- Modular component design — allows systems to be easily upgraded and configured.
- The FP-11 option provides 46 floating point instructions — to make available high-speed floating point math.
- The optional memory management unit provides protection and segmentation for up to 4 megabytes of memory — for efficient use of memory and overall system integrity.

**Table 13-1 LSI-11/23 Configurations**

<b>Model No.</b>	<b>Description</b>	
KDF11-AC		LSI-11/23 CPU without memory management, one dual
KDF11-AA	KDF11-AC+ KTF11-AA	LSI-11/23 CPU with memory management, one dual
KDF11-HK	KDF11-AA, (4) MSV11-DD	LSI-11/23 CPU, 256 KB RAM memory, five duals
KDF11-LK	KDF11-AA, MSV11- LK	LSI-11/23 CPU, 256 KB of parity memory, 2 duals
KEF11-AA	KEF11-AA	Floating-point option for a KDF11-AA, one forty pin package for installation on a KDF11-AA
KTF11-AA		Memory management option for LSI-11/23 CPU, one forty-pin package

**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 loads
Environmental Storage	−40° C to 65° C (−40° F to 149° F) 10% to 90% relative humidity, noncondensing
Operating Temperature	5° C to 60° C, (41° F to 140° F) Maximum outlet temperature rise of 5° C (9° F) above 60° C (140° F)
Altitude	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)	
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)

**ADDITIONAL SOURCES OF LSI-11/23 DOCUMENTATION**

Appendix G of this Handbook lists additional documentation available for the LSI-11/23. Appendix B lists LSI-11/23 timing.

**CONFIGURATION DATA****JUMPER SELECTION**

Several jumpers on the processor module provide user-selectable features. Table 13-2 lists the jumper configurations. Figure 13-1 illustrates the Rev A board jumper locations and Figure 13-2 illustrates the Rev C board jumper locations. Jumpers not discussed are reserved for use by DIGITAL and should not be used.

**Table 13-2 Jumper Configurations**

<b>Jumper</b>	<b>Name</b>	<b>In</b>	<b>Out</b>
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 <sub>8</sub> 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 <sub>8</sub>	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	Wake-up Circuit	Disabled	Enabled

**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.

**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<sub>8</sub>). 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.

**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. Table 13-3 lists power-up mode console print out.

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

This mode causes the microcode to fetch the contents of memory locations 24<sub>8</sub> and 26<sub>8</sub> 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.

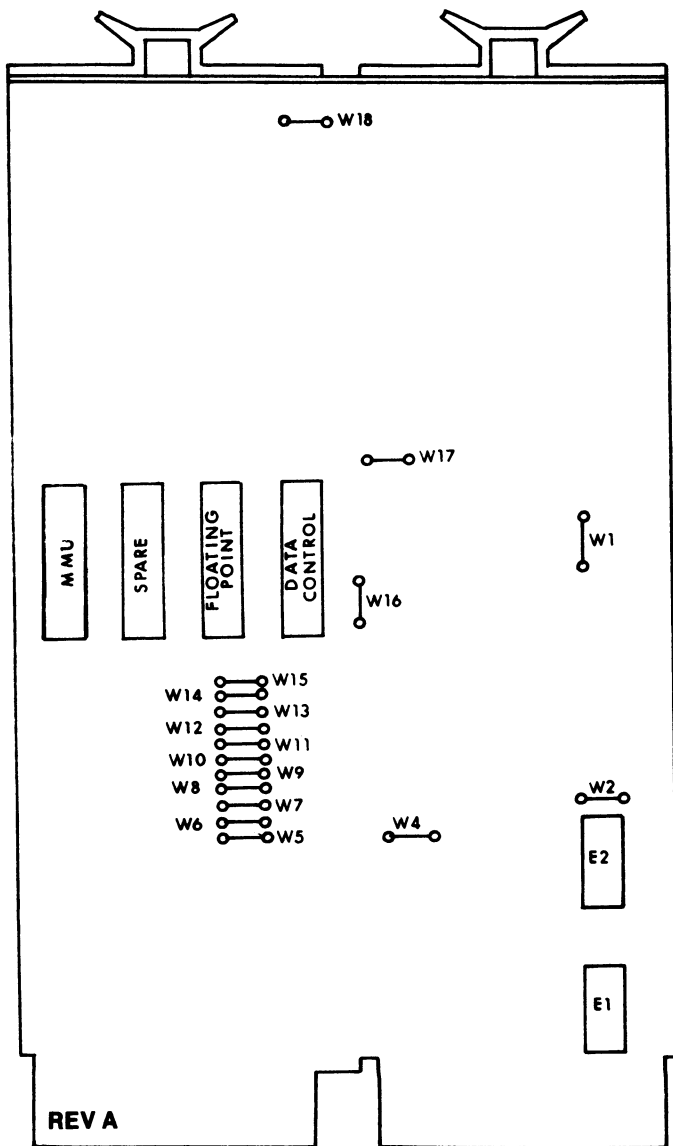


Figure 13-1 LSI-11/23 Jumper Locations (Rev. A)

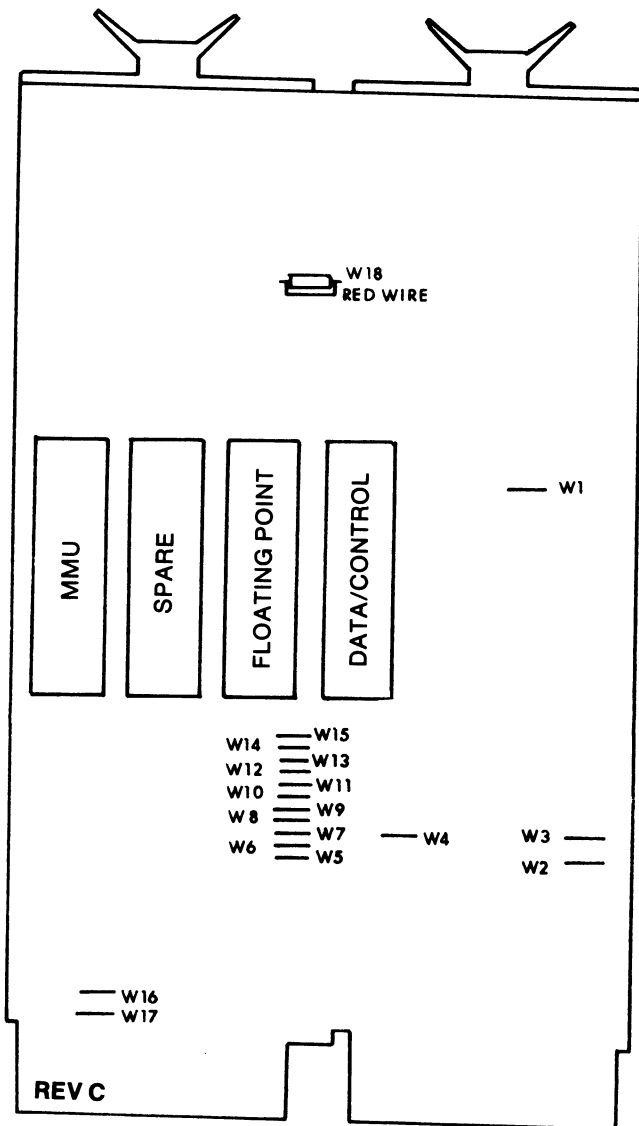


Figure 13-2 LSI-11/23 Jumper Locations (Rev. C)

Table 13-3 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 BDV11, MXV11-A2 options. If jumpers W15 through W9 are used, that address will be printed.
3. The terminal printout will consist of 6 octal digits as specified in the table, followed by a carriage return, line feed, and "@" prompt character in all cases.



### **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.

### **Power-Up Mode 3 (User 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<sub>8</sub> (conventional starting address for DIGITAL systems). If W8 is removed, a trap to 4<sub>8</sub> (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<sub>8</sub>, 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.

### **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 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 except the RKV11 option. 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.

## DESCRIPTION

### Introduction

The LSI-11/23 processor is implemented using two MOS/LSI chips, data and control. The memory management unit (MMU), an optional chip, provides a PDP-11/34 software-compatible memory management scheme. Also available for the LSI-11/23 are two options providing the PDP-11/34 software-compatible FP-11 instruction set.

### Data Chip

The data chip (DC302) performs all arithmetic and logical functions, handles data and address transfers with the LSI-11 Bus (except relocation, which is handled by the MMU) to the external world, and coordinates and generates most of the signals used for interchip communication and external system control. 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.

### Control Chip

The control chip (DC303) implements 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. Figure 13-3 contains the processor functional block diagram.

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 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. Figure 13-4 shows the pictorial layout of the data and control chip.

**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<sub>8</sub>.

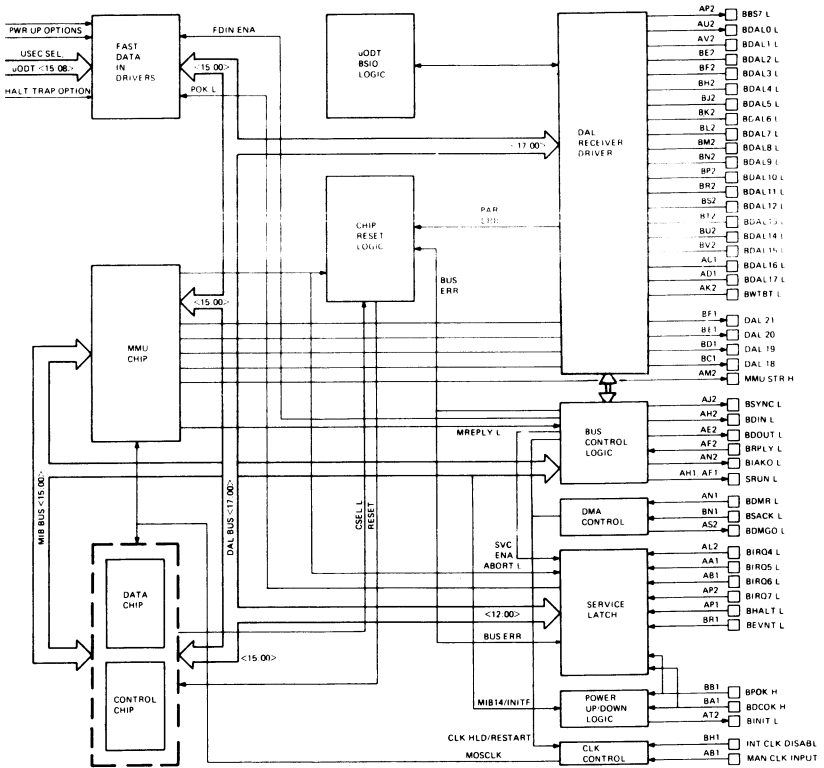


Figure 13-3 Processor Functional Block Diagram

**Processor Status Word (PS)**

The processor status word (PS) is in the data chip and contains information on the current processor status. On the LSI-11/23, the high byte contains additional information not required by the LSI-11, LSI-11/2, and SBC-11/21. As Figure 13-5 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 priori-

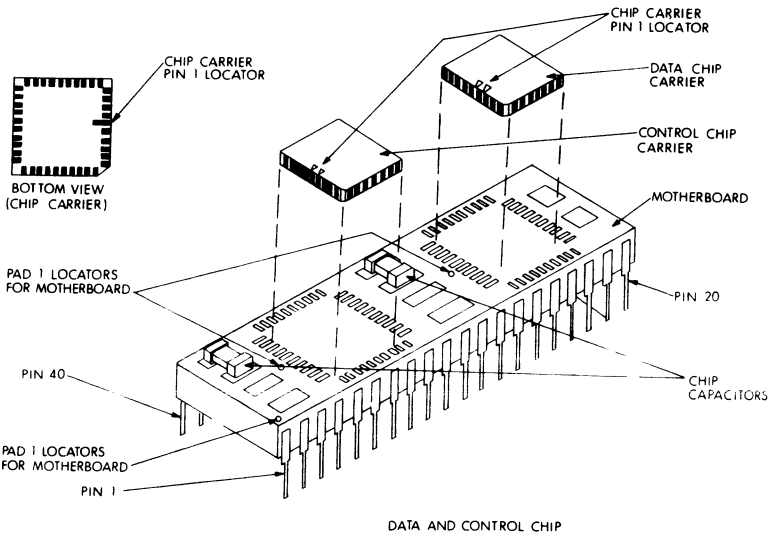


Figure 13-4 LSI-11/23 Data and Control Chip

ty, an indicator of the previous memory management mode, and an indicator of the current memory management mode. Figure 13-5 illustrates the processor status word.

**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).

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

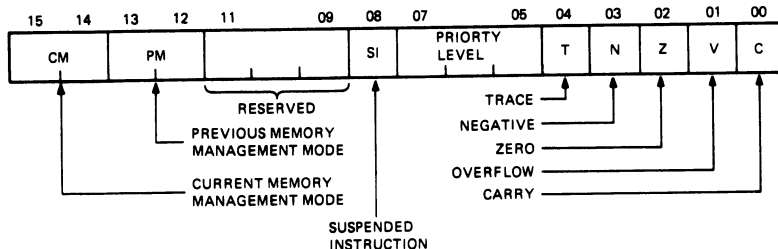


Figure 13-5 Processor Status Word (PS)

**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 management 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.

### 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.

**MIB 15/Memory Management Enable (MME)** — During clock-high time, MIB 15 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 micro-cycle 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.

**MIB 14/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.

**MIB 13/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 Figure 13-6. The PROM decodes them to determine the type of microcycle currently executing within the MOS chips. The PROM outputs 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.

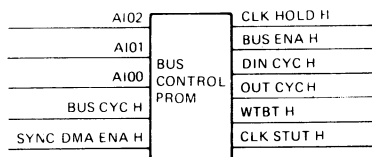


Figure 13-6 Bus Control PROM

**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 BRPLY 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 BRPLY 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.

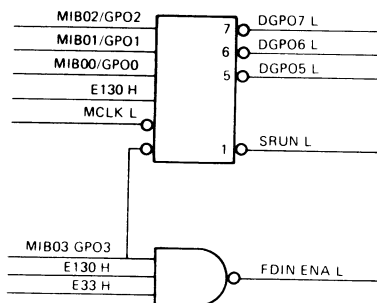


Figure 13-7 GPO Decode Logic

**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 illustrated in Figure 13-7. The decoded output is shown in Table 13-4, General-Purpose Output Signals.

Table 13-4 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

GP02	GP01	GP00	Output Name	Function
				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 event 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.

GP02	GP01	GP00	Output Name	Function
				This signal can be used to cause a steady RUN indication while the processor is executing micro-instructions and a flashing indication when typing characters in console-ODT.

### BSYNC Logic

Figure 13-8, Bus Sync Logic, 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.

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-11 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 BRPLY L, to clear BSYNC L after the rising edge of BRPLY L. 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 LSI-11/23 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 about 12 to 1 over regular programmed transfers is gained by this technique.

The signals required for the DMA logic are the following.

- |                |   |
|----------------|---|
| <b>BDMR L</b>  | 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.  |
| <b>BDMGO L</b> | 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. |

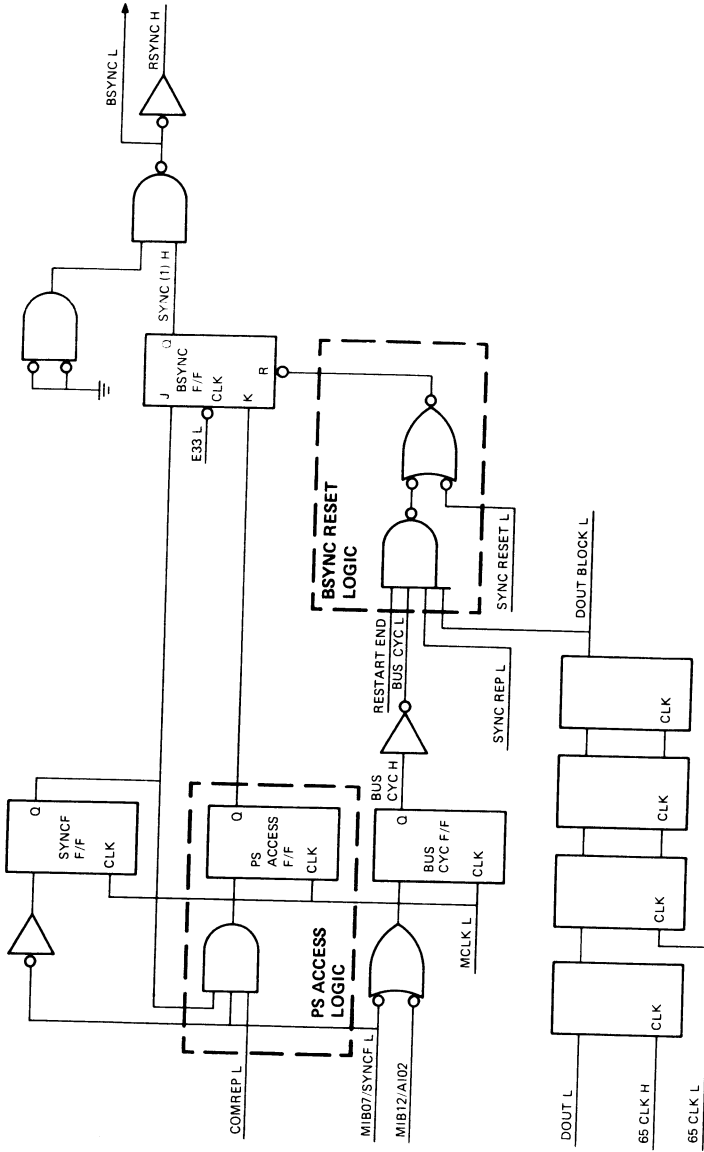


Figure 13-8 BSYNC Logic

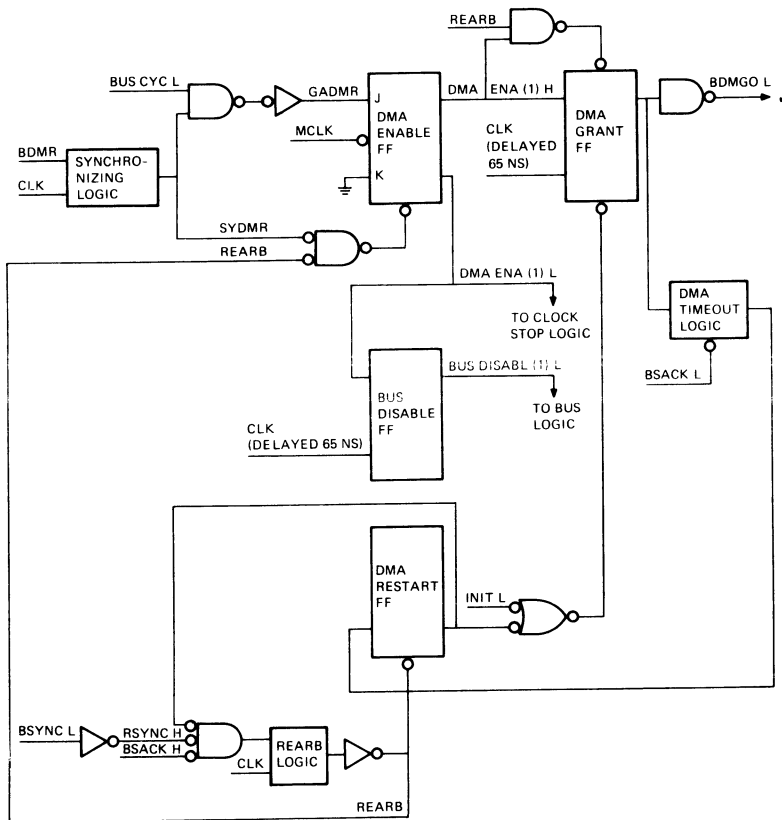


Figure 13-9 DMA Logic

### 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 example 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 LSI-11/23 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 Figure 13-10 is used with a high-frequency crystal clock input to generate the required clock 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 the state until BDCOK H is asserted by the power supply (or the wake-up circuit). The initialization circuitry is shown in Figure 13-11. 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.

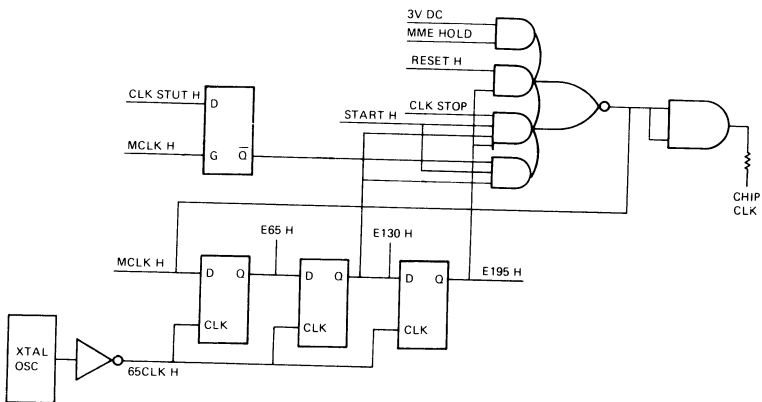


Figure 13-10 Clock Generator

### Wake-Up Circuit

The "wake-up" circuit on the LSI-11/23 (KDF11-AC) module consists of a diode, a resistor, a capacitor, and a Schmitt trigger inverter, also shown in Figure 13-11. 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 no 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 BDOU 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.

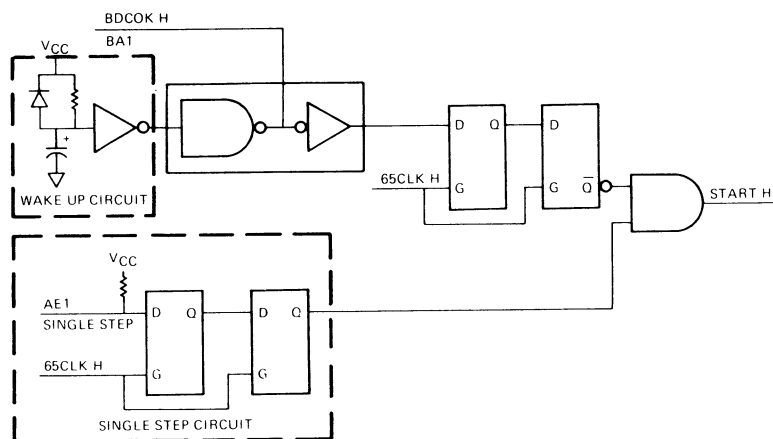


Figure 13-11 Clock Generator Initialization Circuitry



### 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. Figure 13-12 illustrates 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.

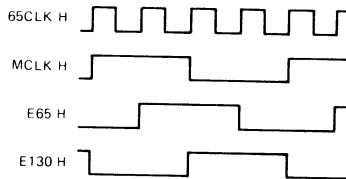


Figure 13-12 Normal Clock Cycle

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 Figure 13-13.

#### 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

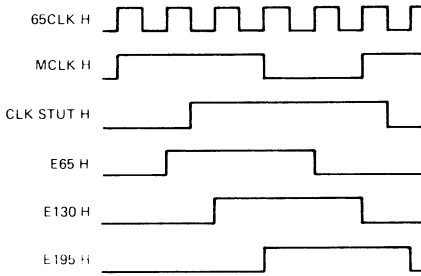


Figure 13-13 Clock Stutter Cycle

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 master-ship back to the processor. The cycle is shown in Figure 13-14.

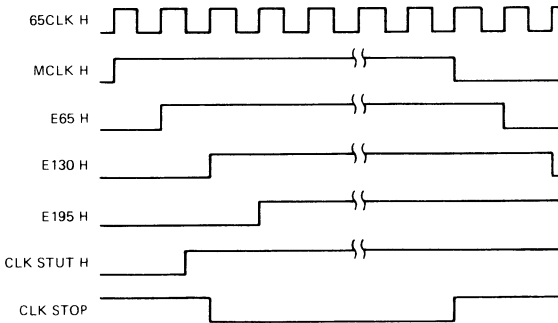


Figure 13-14 Clock Stop Cycle

### Memory Management Cycle

This cycle occurs during address microcycles when the memory management chip is present and is enabled to do address relocation (en-

abling 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 micro-cycle. The relocation circuit, shown in Figure 13-15 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 and 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 Figure 13-16 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.

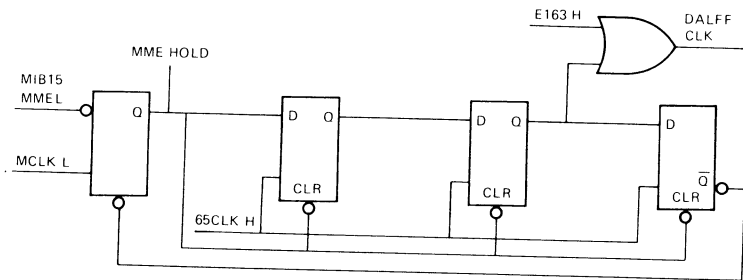


Figure 13-15 Relocation Timing Circuit

### Chip 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.

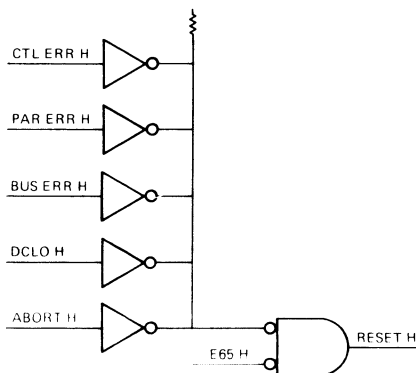


Figure 13-16 Reset Circuit

1. Control error — Nonexistent control chip selected by the microcode. A trap to location  $10_8$  occurs.
2. Bus error — Nonexistent memory location accessed. A trap to location  $4_8$  occurs.
3. Parity error — A parity error detected on a current read from memory. A trap to location  $114_8$  occurs.
4. MMU abort — The MMU has aborted a mapped reference. A trap to location  $250_8$  occurs 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.

## LSI-11/23 PROCESSOR OPTIONS

### KEF11-AA

Forty-six floating point instructions are available as a microcode option (KEF11-AA) on the LSI-11/23 processor. (The MMU option must also be present). 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 rou-

tines and provides for both single-precision (32-bit) and double-precision (64-bit) operands. It 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-11/34A, PDP-11/60, and PDP-11/70. For a complete description refer to Chapter 5.

### **FPF11**

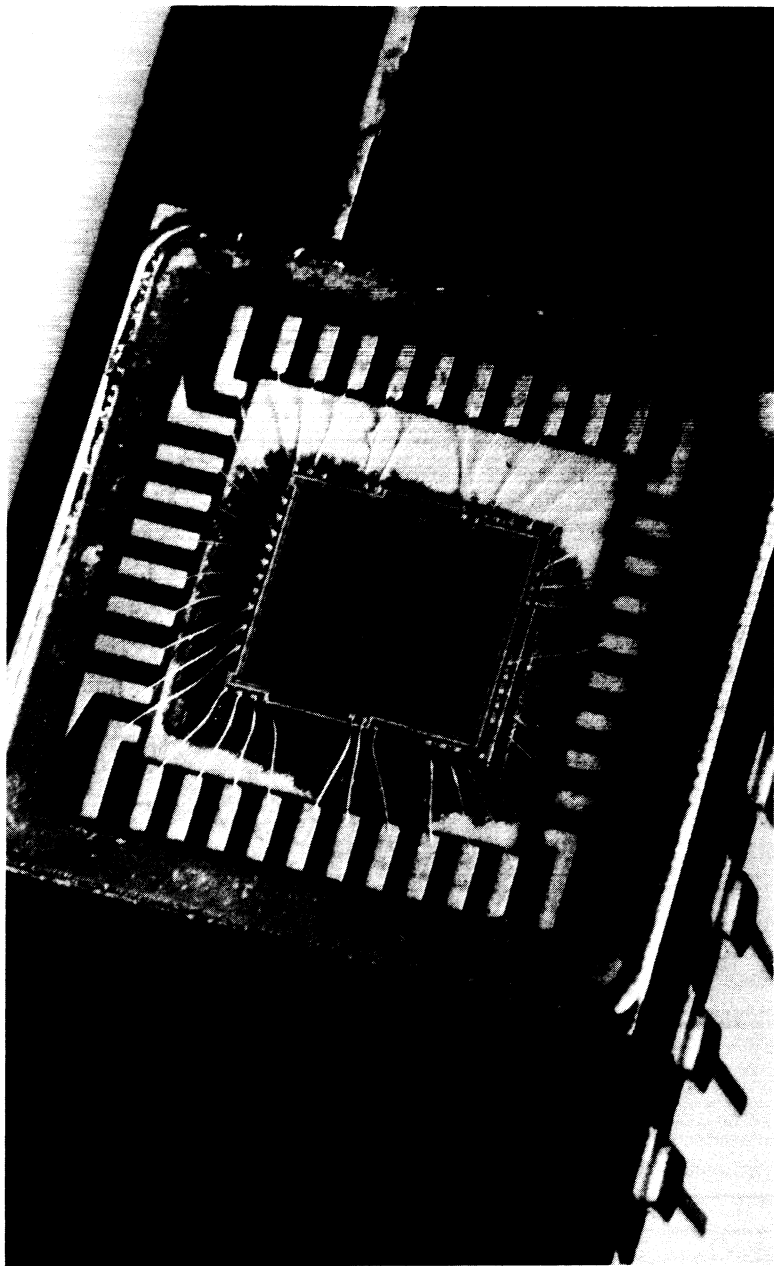
The FPF11 is a hardware implementation of the full PDP-11/34 floating point instruction set. It complements the KEF11-AA, but has six times the improvement in floating-point performance as the KEF11-AA. The FPF11 is a single quad-height module. The MMU is not necessary when utilizing the FPF11.

### **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 for the KEF11 option. This chip provides dual mode (user and kernel) address relocation of 22 bits. Sixteen-bit virtual addresses are received from the data chip via the data/address lines (DAL), relocated to the appropriate 22-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 of the KEF11 option.

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 LSI-11/23 can operate without the MMU chip; however, the memory would be limited to 64 KB and the floating point registers would not be available for the KEF11 option.



**LSI-11 Chip**

## CHAPTER 14

# LSI-11/2 MICROCOMPUTER

### INTRODUCTION

The LSI-11/2 is a 16-bit microcomputer with the speed and instruction set of a minicomputer. Because of its small size (only 5.2 in × 8.5 in, or 13 cm × 22 cm) and unique capabilities, it can fit into almost any instrumentation, data processing, or controller configuration.

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 — BENEFITS

- No on-board memory — flexibility to match RAM/ROM size to requirements.
- Compact, double-height module size — allows for versatile packaging.
- ODT console emulator — ease of program debugging.
- Direct addressing of 32K 16-bit words or 64K 8-bit bytes (K = 1024) — provides flexibility in defining data structures.
- Over 70 operation codes — provide powerful and convenient programming.
- 8 addressing modes for specifying operands — allow for absolute, deferred, autoincrement, autodecrement, and index register references.
- 8 internal general-purpose registers for use as accumulators and for operand addressing — provide flexible programming techniques.
- Stack processing — creates convenient handling of structured data, subroutines, and interrupts.
- Byte-oriented instructions — provide efficient processing of 8-bit characters without the need to rotate, swap, or mask.
- LSI-11 Bus structure — provides 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 speeds.

- Direct memory access (DMA) — allows peripherals to access memory without interrupting processor operation.
- Vectored interrupts — provide fast interrupt response without device polling.
- Power-fail and automatic restart hardware — detects and protects against ac power fluctuations.
- Modular component design — allows systems to be configured and upgraded easily.
- Extended Instruction Set (EIS) and Floating Point Instruction Set (FIS) available as an option — provide fixed and floating point hardware arithmetic.

### 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 H)
Interrupt Latency	35.05 microseconds ±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)
Operating Temperature:	5° C to 60° C (41° to 140° F) Derate the maximum temperature by one degree Celsius for each 1000 feet of altitude above 8000 feet.
Relative Humidity	10% to 90%, noncondensing
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.

**Storage**

Temperature	-40°C to 65°C (-40°F to 149°F)
Relative Humidity	10% to 90%, noncondensing
Altitude	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.

**ADDITIONAL SOURCES OF LSI-11/2 DOCUMENTATION**

Appendix G of this Handbook lists additional documentation available for the LSI-11/2.

**CONFIGURATION DATA**

The LSI-11/2 processor (KD11-HA) is a double-height module, 5 ½" × 8 ½" (13.3 cm × 22.8 cm). Table 14-1 includes current LSI-11/2 configurations.

**Table 14-1 LSI-11/2 Configurations**

<b>Model No.</b>	<b>Board No.</b>	<b>Description</b>
KD11-HA	M7270	LSI-11/2 processor module only (no memory)
KD11-GF	M7270 M8047-AA	LSI-11/2 CPU plus 8 KB multi-function board
KD11-GC	M7270 M8047-CA	LSI-11/2 CPU plus 32 KB multi-function board
KD11-HF	M7270 M8044-A	LSI-11/2 processor module plus double-height MSV11-DA 4K × 16-bit read/write memory module

**JUMPER SELECTION**

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-selectable functions. The factory-configured functions selected are listed in Table 14-2 and illustrated in Figure 14-1. Processor functions may be altered by installing or removing these jumpers.

Processor module etch revisions can be determined by examining the printed circuit board part number on side 2 (solder side) of the processor module.

**Table 14-2 LSI-11/2 M7270 Processor Module Factory-Installed Jumpers**

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

**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 and W4.

**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.

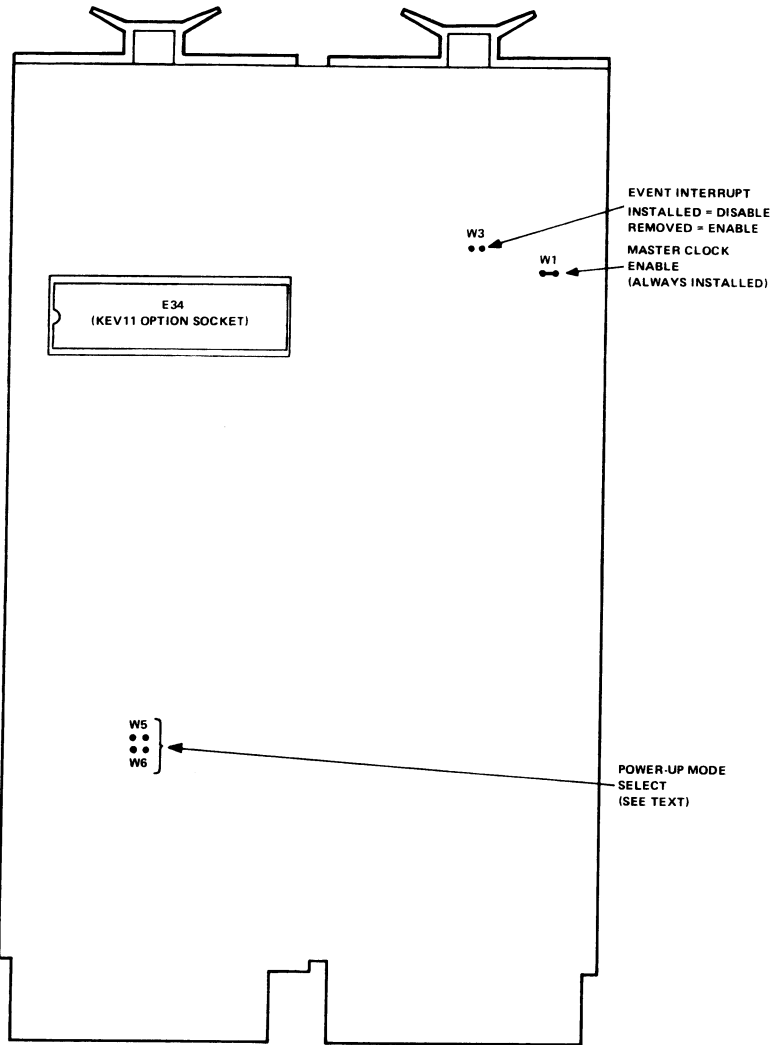


Figure 14-1 M7270 Processor Module Jumper Locations

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 in Table 14-3. Detailed descriptions of each mode are provided in the paragraphs that follow.

**Table 14-3 Power-up Jumpers**

<b>Mode</b>	<b>W6</b>	<b>W5</b>	<b>Mode Selected</b>
0	R	R	PC at 24 and PS at 26, or Halt mode
1	R	I	ODT micro-code
2	I	R	PC at 173000 for user bootstrap
3	I	I	Special processor micro-code (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. Mode 0 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 Figure 14-2.

### **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.

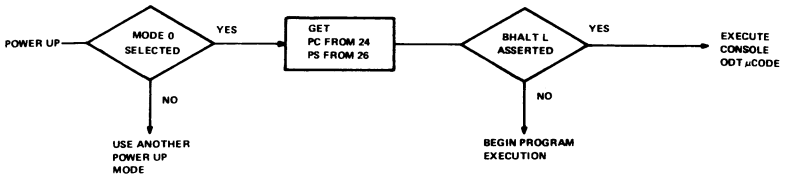


Figure 14-2 Mode 0 Power-Up Sequence

### 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 illustrated in Figure 14-3.

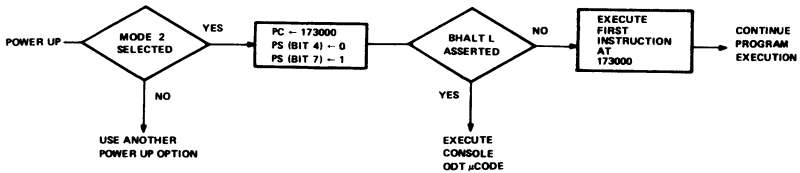


Figure 14-3 Mode 2 Power-Up Sequence

### 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 micro-trap 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  $\mu$ s from the time BEVNT L is asserted; however, if optional EIS/FIS instructions are being executed, this time could extend to 44.1  $\mu$ s maximum. This time could also be extended by processor trap execution (T bit, power-fail, etc.), or by asserting the BHALT L signal.

## DESCRIPTION

### Introduction

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).

### **Data 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.

### **Control Chip**

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. (These functional units are illustrated in Figure 14-4.)

### **CAUTION**

Do not remove processor chips from their sockets. Improper handling will permanently 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 Figure 14-5. 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.

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.

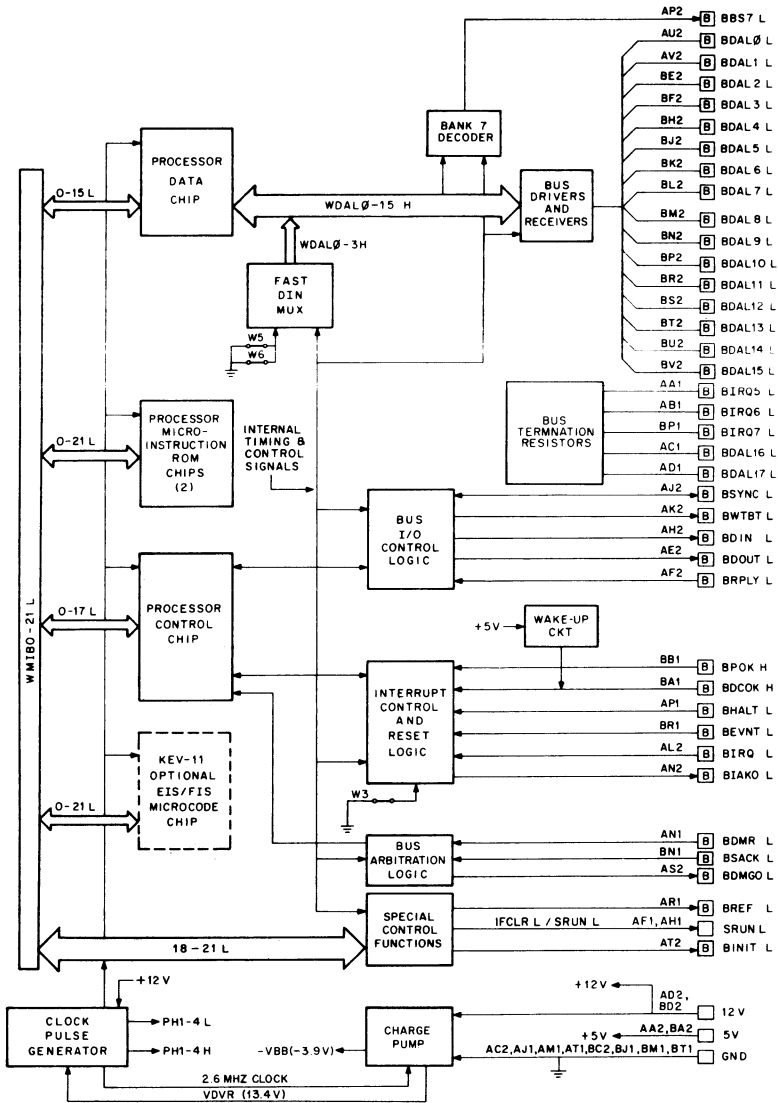


Figure 14-4 M7270 LSI-11/2 Processor Module—Basic Functions



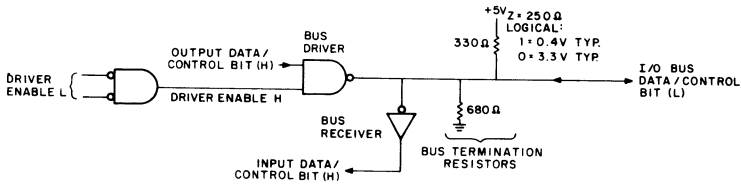


Figure 14-5 LSI-11 Bus Loading and Driver/Receiver Interface

Bus driver output signals and their respective enable signals are listed below:

Bus Driver (Signal)	Enable Signal(s) (Low = Enable)
BSYNC L	
BBS7 L	INIT (1) H, DMGCY H
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 is distributed on the processor module via the WDAL <0:15> H and DAL <0:15> H 16-bit busses. WDAL <0:15> H interface directly with the microprocessor data chip, the DIGITAL 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 used only during the interrupt sequence, it is discussed later. Logic circuits which produce and/or distribute these signals are shown in

Figure 14-6. 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.

**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 from the LSI-11 Bus 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

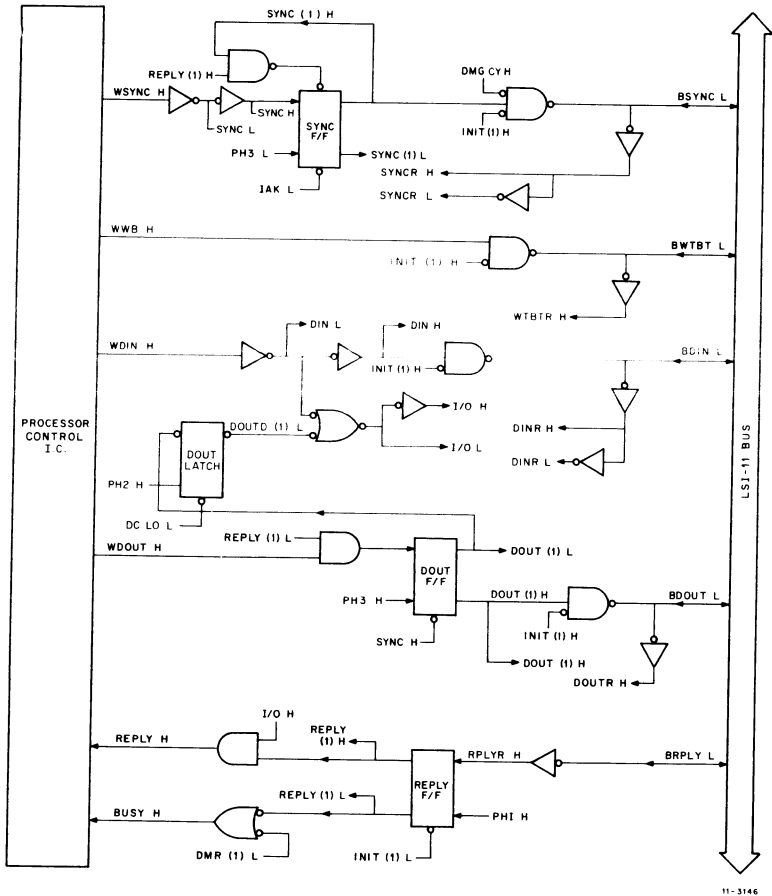


Figure 14-6 Bus I/O Control Signal Logic

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 illustrated in Figure 14-7. 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.

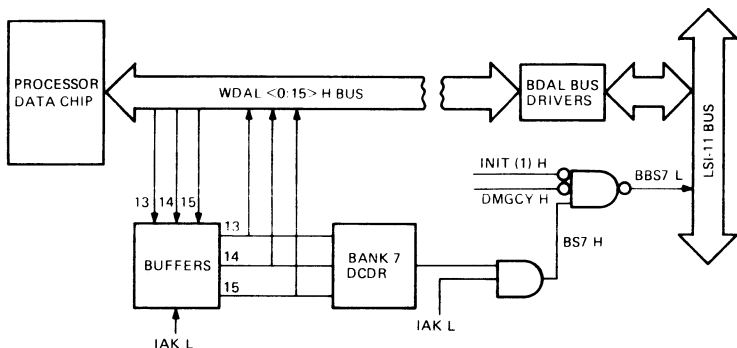


Figure 14-7 Bank 7 Decoder

### Interrupt Control and Reset Logic

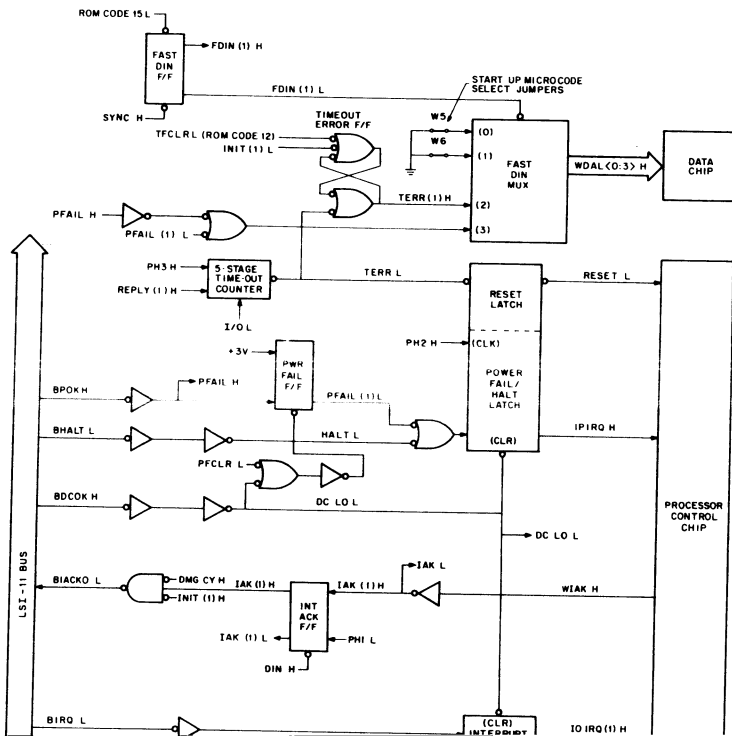
Interrupt control and reset logic functions are illustrated in Figure 14-8. Reset functions include bus error and power-fail (BDCOK H negated). Interrupt functions include power-fail (impending), Halt mode (console microcode control), event (or line time clock) interrupt, and external BIRQ interrupts.

### 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. The processor pushes the PC and PS onto the stack and enters a power-fail routine via vector location 24<sub>8</sub>. The end of this routine must be terminated by a HALT instruction to avoid possible memory corruption. 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

# Chapter 14 — LSI-11/2 Microcomputer



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 LSI-11/2 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 arbitrate DMA requests, and ignore all external interrupts.

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

1. When the BHALT L signal is asserted
2. When a HALT instruction has been executed
3. By a 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

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 DBOU L signal by not returning a BRPLY L signal within 10  $\mu$ s (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  $\mu$ s, the full count (32<sub>10</sub>) 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.

### **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 BMT, BPT, IOT, and TRAP instructions, and hardware-generated trace trap, bus error, power-fail, etc., are described in Chapter 8.

### **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.

### Event Line Interrupt

The event line interrupt function can be used as a line time clock interrupt, or as desired by the user. 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. 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. The interrupt is initiated by the external device by asserting BEVNT L. All other external interrupts are requested by a device's asserting the BIRQ signal. 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<sub>8</sub>, 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 operations and five special control signals which are internal to the processor module. Special control function logic circuits are illustrated in Figure 14-9. Microinstruction 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.

**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.



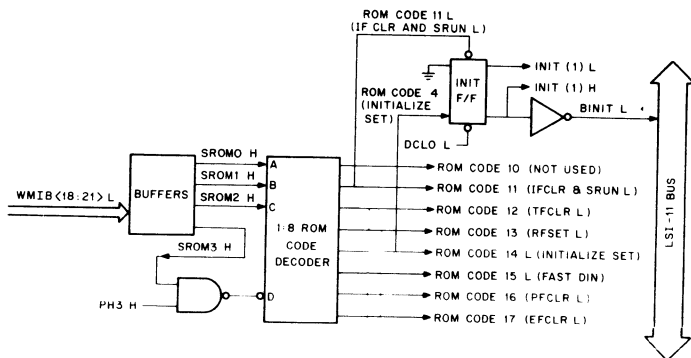


Figure 14-9 Special Control Functions

**ROM Code 12 [TFCLR L]** — This code is a trap function clear signal which clears time-out error flip-flops.

**ROM Code 13 [RFSET L]** — Memory refresh is not provided by the LSI-11/2.

**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  $\mu$ s 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).

### Clock Pulse and Charge Pump Circuits

The clock pulse and charge pump circuits are illustrated in Figure 14-10. 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.

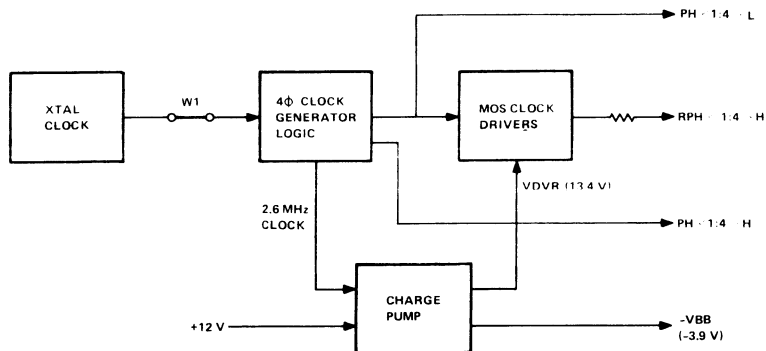


Figure 14-10 M7270 Clock Pulse and Charge Pump Circuits

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  $-V_{BB}$  ( $-3.9$  V) 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 RC circuit receives +5 V 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.

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. The wake-up circuit is illustrated in Figure 14-11.

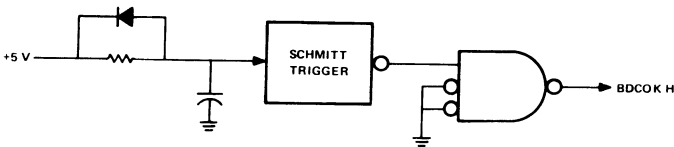


Figure 14-11 M7270 Wake-Up Circuit

### BDAL Bus Driver Enable Logic

The bus driver enable logic is illustrated in Figure 14-12. The four bus driver portions in each of four DIGITAL 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).

### DMA Arbitration Logic

The DMA arbitration logic circuit used on the M7270 processor is shown in Figure 14-13. 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.

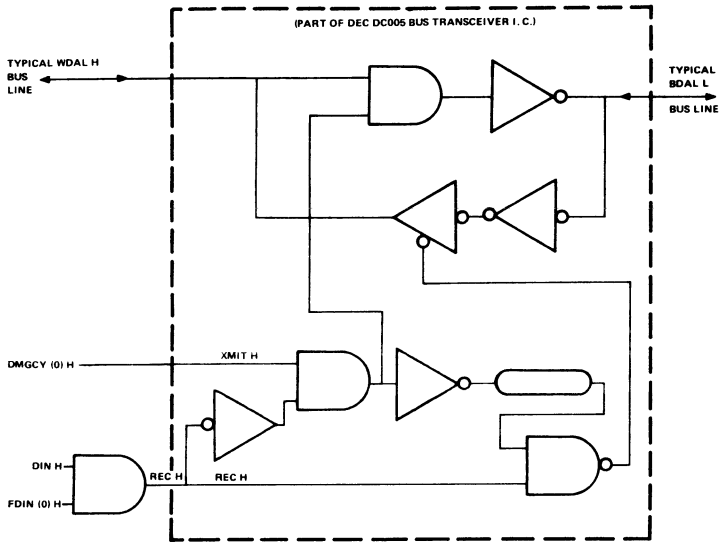


Figure 14-12 M7270 BDAL Bus Driver Enable Logic

## PROCESSOR OPTIONS

### KEV11

The KEV11, an optional microinstruction ROM chip, contains the EIS (Extended Instruction Set) and FIS (Floating Point Instruction Set) microcode on a 40-pin I.C.

The FIS instructions offer floating point addition, subtraction, multiplication, and division, while the EIS instructions permit hardware integer, multiply, and divide.

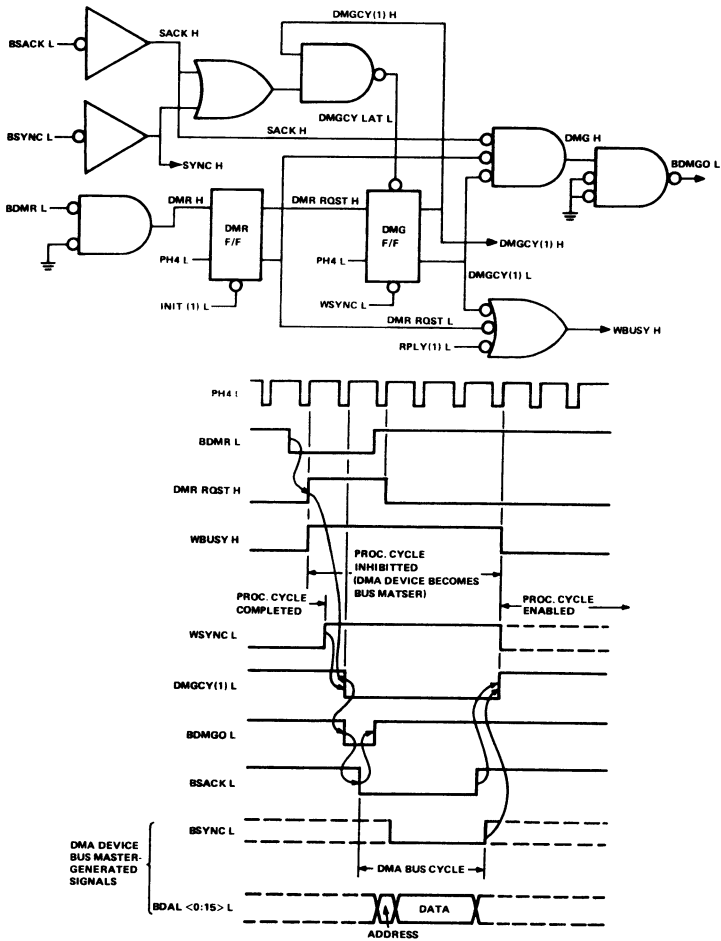


Figure 14-13 M7270 DMA Arbitration Logic and Sequence



## CHAPTER 15

# LSI-11 MICROCOMPUTER

### INTRODUCTION

The LSI-11 is a 16-bit microcomputer with the speed and instruction set of a minicomputer. Due to its size and unique capabilities, the LSI-11 microcomputer 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 microcomputer 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 — BENEFITS

- Compact, double-height module size — allows for versatile packaging.
- ODT console emulator — ease of program debugging.
- Direct addressing of 32K 16-bit words or 64K 8-bit bytes ( $K = 1024$ ) — provides flexibility in defining data structures.
- Over 70 operation codes — provide powerful and convenient programming.
- 8 addressing modes for specifying operands — allow for absolute, deferred, autoincrement, autodecrement, and index register references.
- 8 internal general-purpose registers for use as accumulators and for operand addressing — provide flexible programming techniques.
- Stack processing — creates convenient handling of structured data, subroutines, and interrupts.
- Byte-oriented instructions — provide efficient processing of 8-bit characters without the need to rotate, swap, or mask.
- LSI-11 Bus structure — provides 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 speeds.
- Direct memory access (DMA) — allows peripherals to access memory without interrupting processor operation.
- Vectored interrupts — provide fast interrupt response without device polling.
- Power-fail and automatic restart hardware — detect and protect against ac power fluctuations.
- Modular component design — allows systems to be configured and upgraded easily.
- Extended Instruction Set (EIS) and Floating Point Instruction Set (FIS) available as an option — provides fixed and floating point hardware arithmetic.

### **SPECIFICATIONS**

Identification	M7264
Size	Quad
Dimensions	26.6 cm × 22.8 cm (10.5 in × 8.9 in)
Power Requirements	+5 V + 5%, 1.8A +12 V +5%, 0.8A
Bus Loads	ac 2.4 unit loads dc 1 unit loads
Instruction Timing	See Appendix C
Interrupt Latency	35.05 microseconds 120% (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)
Operating Temperature	5° C to 60° C (41° to 140° F) Derate the maximum temperature by one degree Celsius for each 1000 feet of altitude above 8000 feet.
Relative Humidity	10% to 90%, noncondensing
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.

Storage Temperature	-40°C to 65°C (-40°F to 149°F)
Relative Humidity	10% to 90%, noncondensing
Altitude	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.

**ADDITIONAL SOURCES OF LSI-11 DOCUMENTATION**

Appendix G of this Handbook lists additional documentation available for the LSI-11.

**CONFIGURATION DATA**

**JUMPER SELECTION**

The LSI-11 processor module is factory-configured for specific functions. In many applications the processor 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 Table 15-1. 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 mod-

ules as shown in Figure 15-1. Jumpers for M7264 and M7264-YA LSI-11 processor-modules are illustrated in Figure 15-2.

**Table 15-1 LSI-11 Processor Module Factory-Installed Jumpers**

Jumper	M7264		M7264-YA	
	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

Jumper	Status	Function	Status	Function
W11	I	Enable on-board memory select	R	Disable on-board memory select

**NOTE**

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

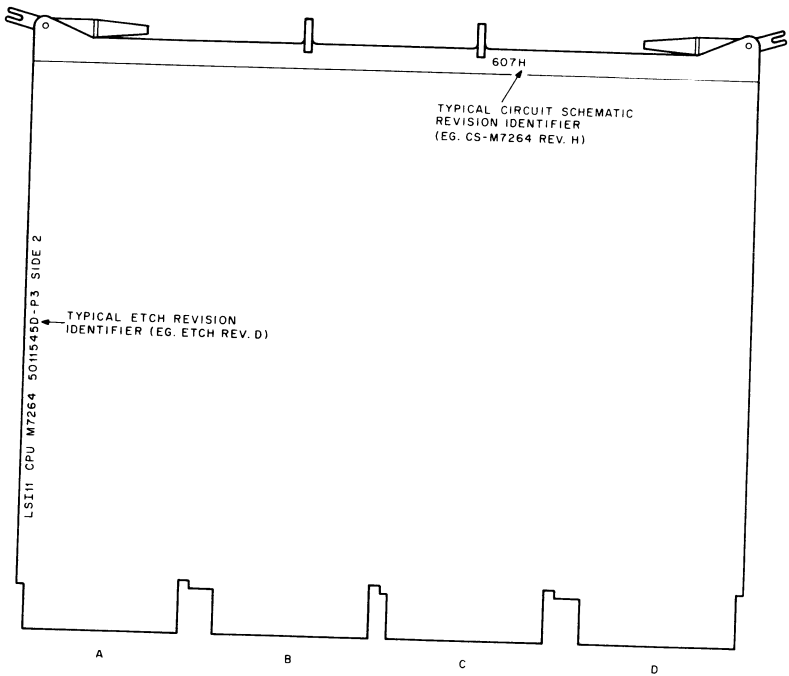


Figure 15-1 Module Etch and Circuit Schematic Revision Identifiers (Module Side 2 Shown)

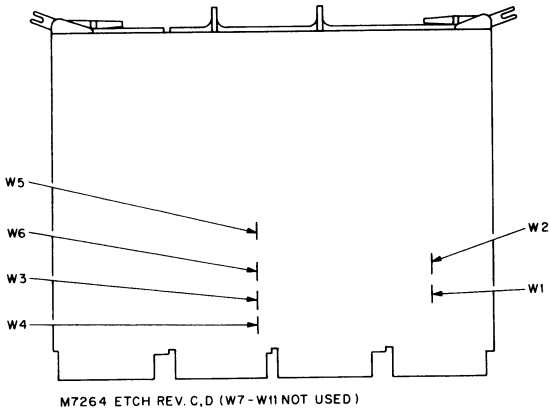
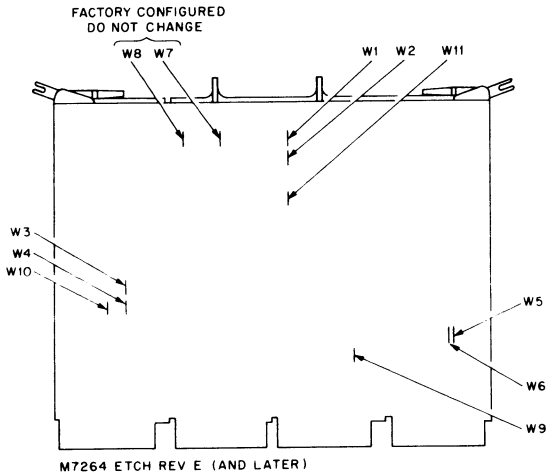


Figure 15-2 M7264 and M7264-YA Processor Module Jumper Locations

**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.

Mode	Jumpers*		Mode Selected
	W6	W5	
0	R	R	PC at 24 and PS at 26, or Halt mode
1	R	I	ODT microcode
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 mode 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. Mode 0 should be used only with nonvolatile memory (or

### Power-Up Mode 3

This mode 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  $\mu$ s from the time BEVNT L is asserted; however, if optional EIS/FIS instructions are being executed, this time could extend to 44.1  $\mu$ s 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 BSYNC L/BDIN L transactions. This process takes approximately 130  $\mu$ s during which external interrupts (BIRQ L and BEVNT L) are ignored. However, DMA requests can be granted between each of the 64 refresh transactions.

## **PROCESSOR OPTIONS**

### **KEV11**

The KEV11, an optional microinstruction ROM chip, contains the EIS (Extended Instruction Set) and FIS (Floating Point Instruction Set) microcode on a 40-pin I.C. The FIS instructions offer floating point addition, subtraction, multiplication, and division, while the EIS instructions permit hardware integer multiply and divide.







## CHAPTER 16

# LSI-11 SYSTEM TROUBLESHOOTING

### INTRODUCTION

The LSI-11 System Troubleshooting Chapter is designed to address troubleshooting procedures for general LSI-11 based microcomputer systems. While this chapter does not attempt to cover problems at the chip level, it does present board-swapping techniques. Before delving into this chapter, it is recommended that you have already a fundamental understanding of the basic LSI-11 concepts, which should have been acquired by reading this Handbook.

This chapter has been constructed to provide a logical sequence to detect and isolate a problem in your system. Flowcharts appear quite frequently to supplement the text, and serve as step-by-step visual guides to help you isolate problems in your system, and determine the proper procedures to correct them.

The hardware tools required to troubleshoot your system must include a CPU (this spare CPU must prove to be in good operating order, having undergone successful, repeated testing, and be immediately available for use), a serial line interface unit, and memory. A volt meter, if available, is very useful because it can be used to monitor power supply voltages. A scope can also be very helpful, but for performing board-swapping troubleshooting, it is not required.

### General Troubleshooting Procedure

A flowchart depicting this procedure appears in Figure 16-1. To begin the troubleshooting procedure, apply system power. The system will respond in one of two ways:

1. System does not respond. An ODT PROMPT does not appear. (Condition One)
2. System responds with an ODT PROMPT displayed on your terminal. The ODT PROMPT appears as an "@" sign. (Condition Two)

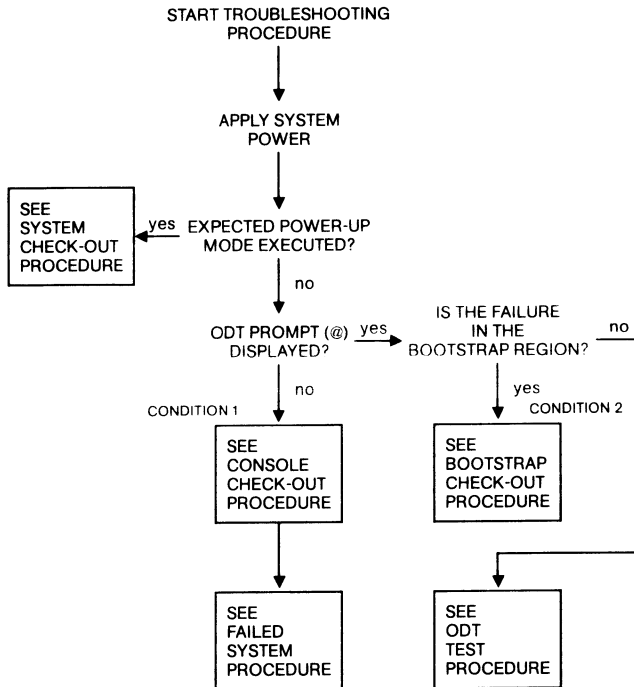


Figure 16-1 General Troubleshooting Procedure

If your system is comprised of new components, follow the Module Installation Procedure prior to applying power. See example below:

### Module Installation Procedure

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 BV

### CAUTION

Do not plug in modules with power applied to backplane.

6. Turn off power.
7. Install modules.

### CONDITION ONE

#### Console Check-Out Procedure

In the first case, where the ODT PROMPT does not appear, the Console Check-Out Procedure should be implemented. (See Figure 16-2). Since there was no response, this may mean that your console is not operating. To determine whether the system is operating, test the cursor action. If cursor action is evident, depress the BREAK key to see if the system is "hung". (BREAK must be configured to halt the system in the SLU interface). If an ODT PROMPT does appear, the system is not "hung" and you can continue with the ODT Test Procedure.

Should there be no response after depressing the BREAK key, the terminal must be tested to check that it is sending characters to the computer. This is accomplished by typing in LOCAL. If the terminal does type in LOCAL, check the console cabling, console serial line unit configuration, the baud rate, the number of stop and data bits---to make sure they match those on the terminal. (The terminal should match what you've configured your board for). In cases where the terminal fails to type in LOCAL, it should be replaced and the Console Check-Out Procedure restarted.

In cases where no cursor action occurs, the console should be checked to determine if it's powered up and on-line. Also, the power connections and cabling should be checked to make sure the fuse hasn't blown. After having accomplished these examinations, and your system still fails to boot/run, proceed immediately to the Failed System Procedure.

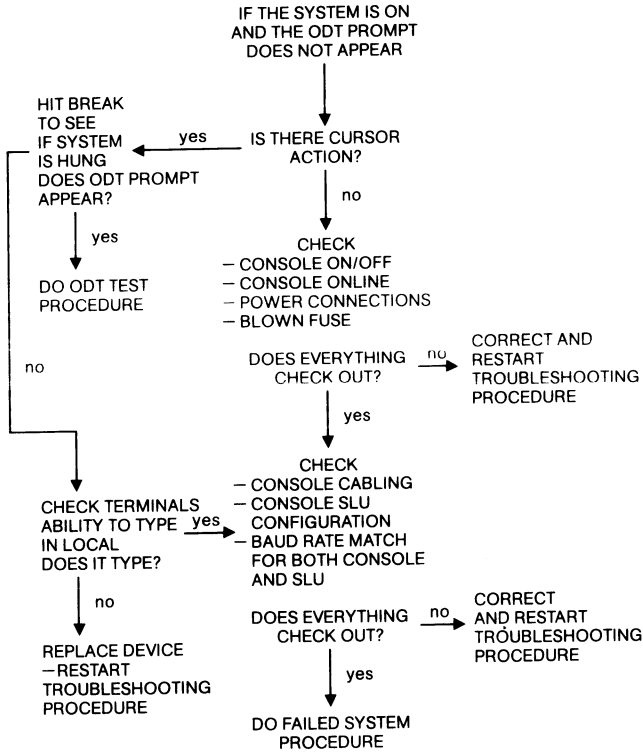


Figure 16-2 Console Check-Out Procedure

**Failed Systems Procedure**

Under the Failed Systems Procedure, (Please refer to Figure 16-3.) you should first determine that memory is configured starting in memory bank zero (0-4K words). Both the LSI-11/23 and the LSI-11/2 microcomputers will not power-up unless memory is configured in bank zero.

If memory is residing in bank zero with no resulting action being generated on the terminal, check the power supply and the voltage levels on the backplane. These voltages should be read as +5v and +12v in accordance with the specifications. If these figures do not check out according to specifications, it means that either the power supply or the backplane is faulty. Should this be the case, perform a swap and restart the Troubleshooting Procedure. If the voltage power levels

agree with the specifications and the system still fails to power-up, check to see if the processor is configured for a valid power-up mode. If so, examine the processor to be sure that the WAKE UP circuit is disabled for use in DIGITAL-supplied boxes and enabled for customer power supplies (except for BA11-VA DIGITAL box).

At this point, if the system still fails to come up, determine whether the memory is self-refreshing, or depending on the processor for refresh. Neither the LSI-11/23 nor the LSI-11/2 provides refresh signals over the bus. However, DIGITAL's earlier LSI-11 microcomputer does.

If the system continues to fail, reconfigure the system to a minimum configuration, consisting of the CPU and console serial line unit, and be sure that memory is in bank zero. Now, try to bring the system up again. If it still doesn't come up, swap the CPU, memory, and serial line unit (one at a time) with tested operating units until the system becomes activated. Once the system is activated, you should perform the ODT Test Procedure.

In most cases, by this time, your system should be operating successfully because the spare parts with which you've just reconfigured your system should diagnose that part of the system that previously failed.

## **CONDITION TWO**

The second failure mode, the case in which the "@" sign or PROMPT is displayed on your terminal after applying system power, should be accompanied by a number. If this number falls between the 173000-173777 range (the bootstrap region), the Bootstrap Check-Out Procedure must be implemented because a failure has occurred in the bootstrap region. If another number prints out on the terminal, and is not in the 173000-173777 range (IE: "@" sign or no number) this indicates that a failure has occurred in another part of memory and you must advance to the ODT Test Procedure to determine where the fault occurred.

### **Bootstrap Check-Out Procedure**

This procedure assumes that the system has halted in the designated bootstrap region, between 173000-173777. (Please refer to Figure 16-4.)

First, determine whether code is residing in this region using ODT. Then, identify the bootstrap device in your system by examining the module numbers on the handle to determine if your bootstrap device is a BDV11, MXV11, or REV11. BDV11 is M8012; MXV11 is M8047; and REV11 is M9400. (The BDV11 bootstrap device board contains very helpful light emitting diodes, LEDs, which by being on or off, can tell you what the failure mode was.) If you still encounter difficulty, note the

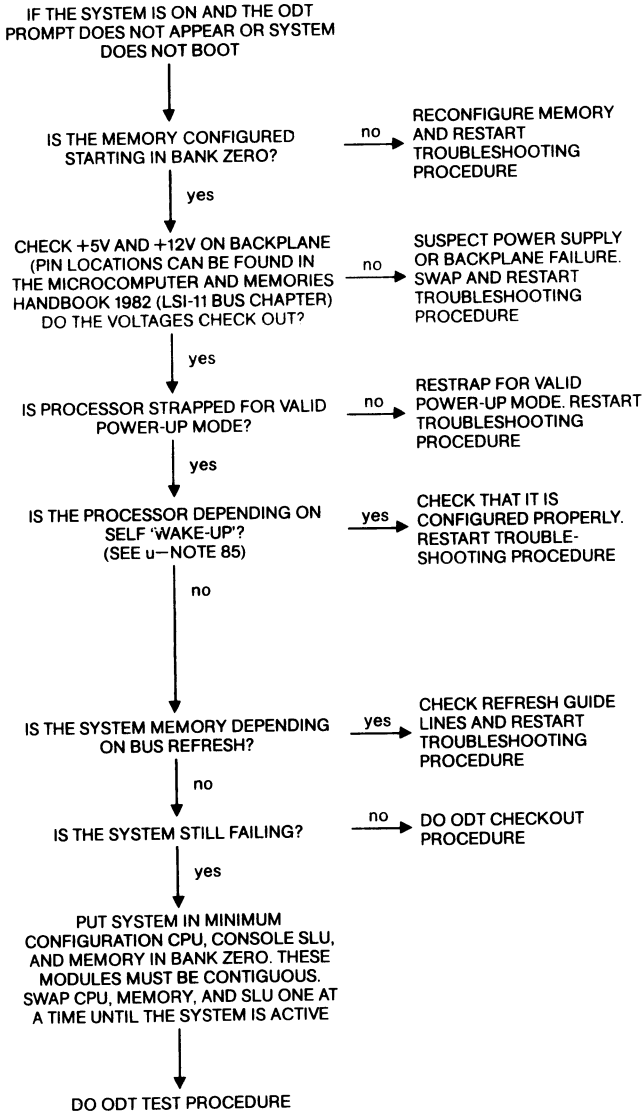


Figure 16-3 Failed Systems Procedure

region number that is printing out with the "@" sign on your terminal, then proceed to the bootstrap listing to check the device you're using. This will tell you where in the bootstrap region the failure occurred. At this point, you can replace either the failing floppy disk or the faulty interface. If the HALT address is not one of the standard bootstrap HALT addresses advance to the ODT Test Procedure. (See Figure 16-5.)

### **ODT Test Procedure**

The ODT Test Procedure should be implemented when the HALT address is not in the bootstrap area or the address is not a standard bootstrap failing address.

To begin, check that the processor is strapped for the correct power-up mode. If so, determine the nature of the failure. On the LSI-11 and the LSI-11/2, the "M" command can provide helpful information to determine how the processor got into the HALT mode. The "M" command is explained in detail in Chapter 7, the ODT chapter of this Handbook. The LSI-11/23 does not have the "M" command. Next, ensure that the modules are contiguous in the backplane and check refresh for the memory. If the memory is being refreshed and all your modules are contiguous in the backplane, remove all the system components except the processor, console SLU and memory in bank zero and restart the system.

At this point in time, you should be able to run the NOP program.

### **No Operation (NOP) Program**

The NOP program is a short program which tests whether the processor can run. Enter the NOP program using ODT starting at location 1000 and type "P" to start. If the program runs, then you can be sure that the processor is at least executing instructions and is not stuck in the HALT mode. To halt the program, depress the console BREAK key or the HALT switch. To restart, type "P".

If your program does not run, then either the processor, the console SLU or the power supply is at fault. To determine which one of these is faulty, perform a module swap. The NOP program is shown below.



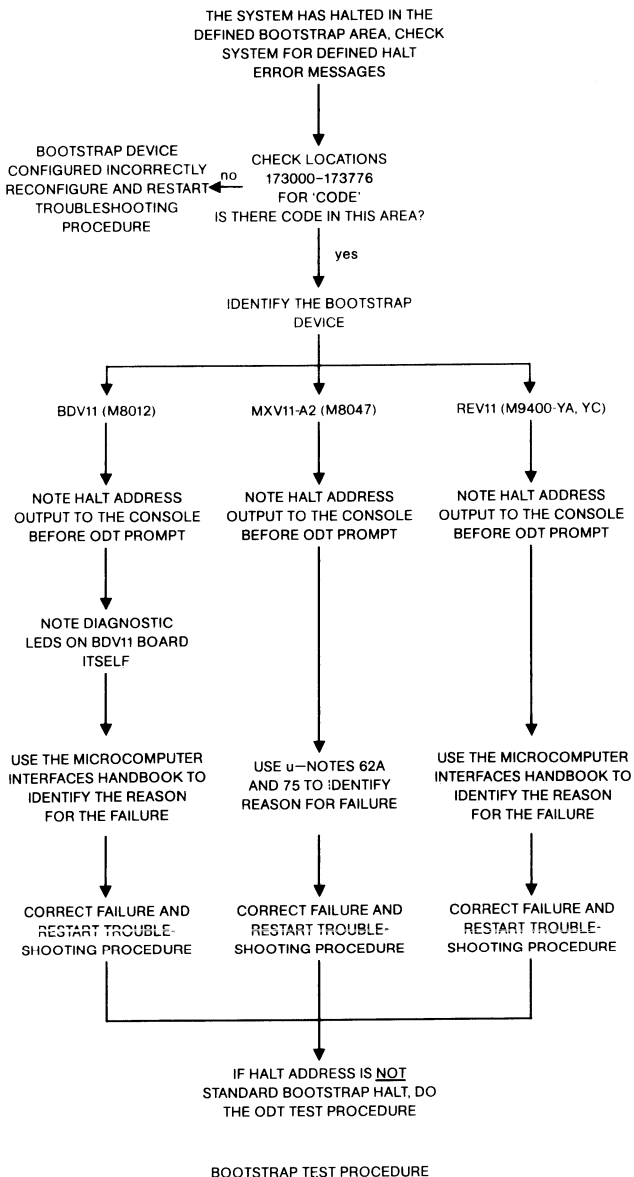


Figure 16-4 Bootstrap Check-Out Procedure

# Chapter 16 — LSI-11 System Troubleshooting

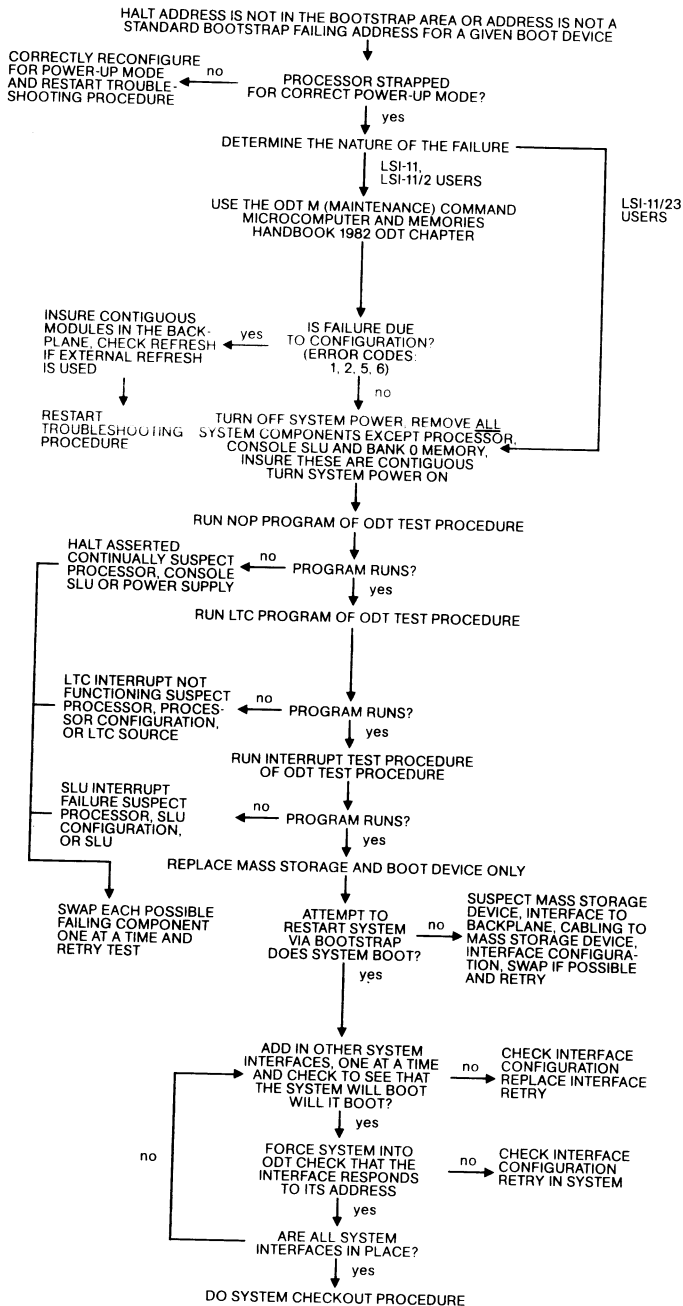


Figure 16-5 ODT Test Procedure

## NOP PROGRAM

ENTER IN ODT

1000/ 240 (LF)

NOP

1002/ 240 (LF)

NOP

1004/ 240 (LF)

NOP

1006/ 137 (LF)

JUMP @ #1000; LOOP FOREVER

1010/1000 (CR)

R7/1000 (CR)

;SET-UP TO DISABLE INTERRUPTS

RS/ 340 (CR)

R6/1000 (CR)

;SET STACK

- TYPE P TO START

- HALT PROGRAM BY USING CONSOLE BREAK OR HALT

SWITCH

- TYPE P TO RESTART

- THIS SIMPLE PROGRAM TESTS TO INSURE THE SYSTEM IS NOT "STUCK" IN HALT MODE

## Line Time Clock (LTC) Program

If the NOP program runs successfully, the line time clock should be tested as well to check that it is running correctly. The LTC program is a short program that is entered under ODT and started by typing "1000 G". (See LTC program below.) The program should halt at location "104" and your terminal will display "106". This indicates that you have received a line time interrupt. If the system does not halt, the line time interrupt is not occurring, meaning that it is not being produced by the power supply or that it is not being recognized by the processor.

## LTC PROGRAM

ENTER IN ODT

100/ 104 (LF)

; BEVNT SERVICE ADDRESS

102/ 340 (LF)

104/ 0 (CR)

; BEVNT SERVICE ROUTINE HALT SYSTEM

```
1000/ 137 (LF)
JMP @ #1000; LOOP ON ITSELF
1002/1000 (CR)
R6/1000 (CR)
; SET STACK
```

- TYPE 1000G
- SHOULD HALT AT 104 AND DISPLAY 106
- IF SYSTEM DOES NOT HALT BEVNT INTERRUPT IS NOT OC-  
CURING

### **Interrupt Test Procedure**

If the LTC program runs successfully, the general interrupt servicing capability of the processor should then be tested. The Interrupt Test Procedure is the third and longer of the three procedures (the others are NOP and LTC) that are entered under ODT. To begin, set up the processor stack so that it can receive interrupts, raise the processor priority, while you enable the interrupt, and enable the interrupt enable bit so that when the serial line unit receives a character, it will interrupt the processor. The Interrupt Test Procedure program is shown below.

When an interrupt occurs, it will vector through locations 60<sub>h</sub> and 62<sub>h</sub>. Location 60<sub>h</sub> contains 2000<sub>h</sub> and points to the receiver service routine. Location 62<sub>h</sub> contains 340<sub>h</sub> which disables future interrupts. The routine resident at location "1000" checks to determine if the transmit buffer is ready to receive a character. If it isn't ready, it waits. If it is ready to receive a character, it takes the character you've just entered in the terminal, retrieves it from the receiver buffer, places it into the transmit buffer, and types it back out onto the console. It then returns from the interrupt. Typing "1000 G" and inputting a character should cause the character to echo right back out to the terminal. You can then hit the BREAK key to halt the test.

The Interrupt Test Procedure also tests the processor's ability to monitor the console serial line unit's capability of generating interrupts and the configuration of the serial line unit.

If your program does not run successfully under the Interrupt Test Procedure, either a SLU failure or a processor failure has occurred.

If your program does run and your system still does not boot, the mass storage device may be causing the failure. You can attempt to restart the system by entering a hand bootstrap found in Appendix A. Also, the interface should be checked, as well as the cabling.

Once your system comes up, is running, and boots, add the devices you removed one at a time to determine which one caused the failure, and make sure they all work.

If the system continues to boot, check under ODT that the device is where you configured it to be by addressing the location of the controlling status register.

Once all system interfaces are back in place and your system is running, perform the System Check-Out Procedure to be certain that the system is running properly.

### INTERRUPT TEST

```
ENTER IN ODT
    060/002000 (LF);                ;RECEIVER SERVICE
ROUTINE
    062/000340 (CR)

    100/000102 (LF);                ;Set LTC INTERRUPT TO
RETURN
    102/000002 (CR)

1000/012706 (LF)                    MOV #1000, SP
; SET UP STACK
1002/001000 (LF)
1004/106427 (LF)                    MTPS #340
; RAISE PRIORITY

                                      WHEN ENABLING
1006/000340 (LF)
; INTERRUPTS
1010/012737 (LF)                    MOV #100, @ #1777560
; SET INTERRUPT ENABLE IN
1012/000100 (LF)
; RECEIVER
1014/177560 (LF)
1016/106427 (LF)                    MTPS #0
; LOWER PRIORITY
1020/000000 (LF)
1022/000137 (LF)                    JMP @ #1022
; WAIT FOR AN INTERRUPT
1024/001022 (CR)
2000/010537 (LF)                    TSTB @ #177564
; IS TRANSMIT READY?
```

```
2002/177564 (LF)
2004/100375 (LF)          BPL .-4
; NO, WAIT
2006/013737 (LF)          MOV @ #177562, @ #
177566
; PUT-INPUT
2010/177562 (LF)
; CHARACTER INTO TRANS
2012/177566 (LF)
; MIT BUFFER
2014/000002 (CR)          RTI
; RETURN FROM INTERRUPT
```

-TYPE 1000G, THEN INPUT A CHARACTER, IT SHOULD ECHO  
RIGHT BACK,  
THEN HIT BREAK TO HALT TEST.

-THIS PROGRAM TESTS THE PROCESSOR'S ABILITY TO HANDLE  
INTERRUPTS,  
THE CONSOLE SLU's ABILITY TO GENERATE INTERRUPTS,  
AND THE CONFIGURATION OF THE SLU.

### **System Check-Out Procedure**

This procedure (See Figure 16-6) should be implemented after the system has been booted to test the complete functionality of each system component. The most effective testing method is to run the diagnostic under XXDP+ for each module individually in the system and run a performance exerciser. You should allow these diagnostics to run overnight so that the intermittent failures in your system can be found. Also, if you're experiencing intermittent problems, running diagnostics and performance exercisers assures that the problem can be located in your system.

If each diagnostic passes, your system has proved that it is running well. If not, replace the faulty component and run the diagnostics overnight again.

### **RELATED MICROCOMPUTER AIDS**

The 1982 Microcomputers and Memories Handbook is very useful and provides informative technical detail, as well as reference material. The Micronotes are also very helpful in solving any further problems you may be experiencing that are not covered in this chapter. If you feel you need to contact someone directly about a problem, call your local sales office for referral.

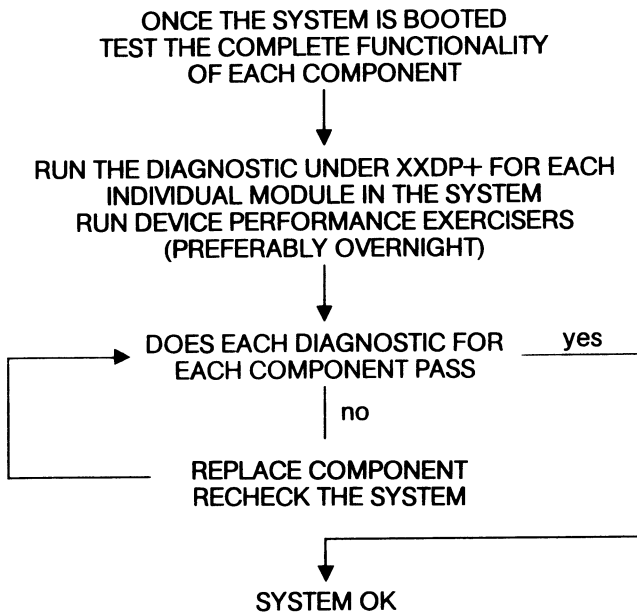
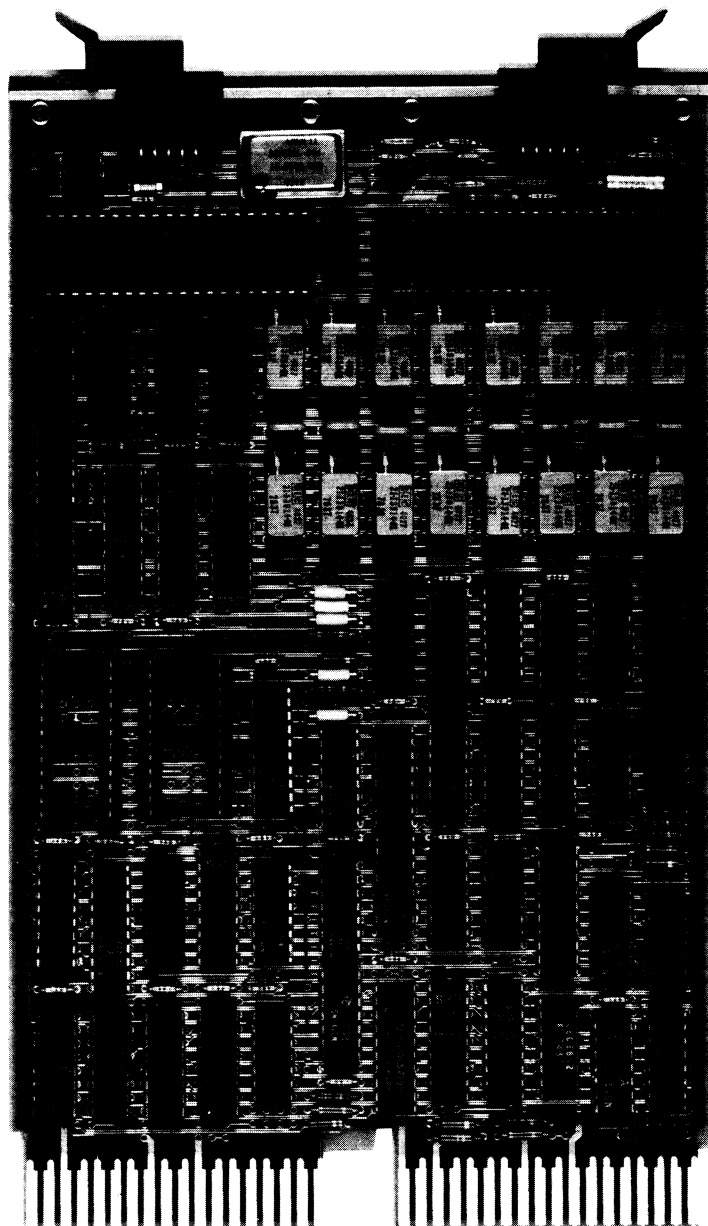


Figure 16-6 System Check-Out Procedure







## CHAPTER 17

# ROM MEMORIES

### MRV11-AA 4K BY 16-BIT READ-ONLY MEMORY

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 — BENEFITS

- Using 512 by 4-bit chips — yields 4096 by 16-bit capacity.
- Using 256 by 4-bit chips — yields 2048 by 16-bit capacity.
- Compatibility — uses chips available from multiple sources.
- User-configured 4K address — allows easy memory layout.

#### SPECIFICATIONS

Identification	M7942
Size	Double
Power	
4K × 16 ROM less PROM integrated circuits	+5V ±5% at 0.4 A
32 512 × 4 PROM integrated circuits	+5V ±5% at 2.8 A
Bus Loads	
AC	1.8
DC	1.0

#### CONFIGURATION

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 an 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 17-1.

**Table 17-1 MRV11-AA PROM Types**

<b>Manufacturer or Source</b>	<b>512 by 4-Bit PROMs</b>	<b>256 by 4-Bit PROMs</b>
Digital	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, 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 17-1.

### **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 17-2; use data listed for the PROM type being used.

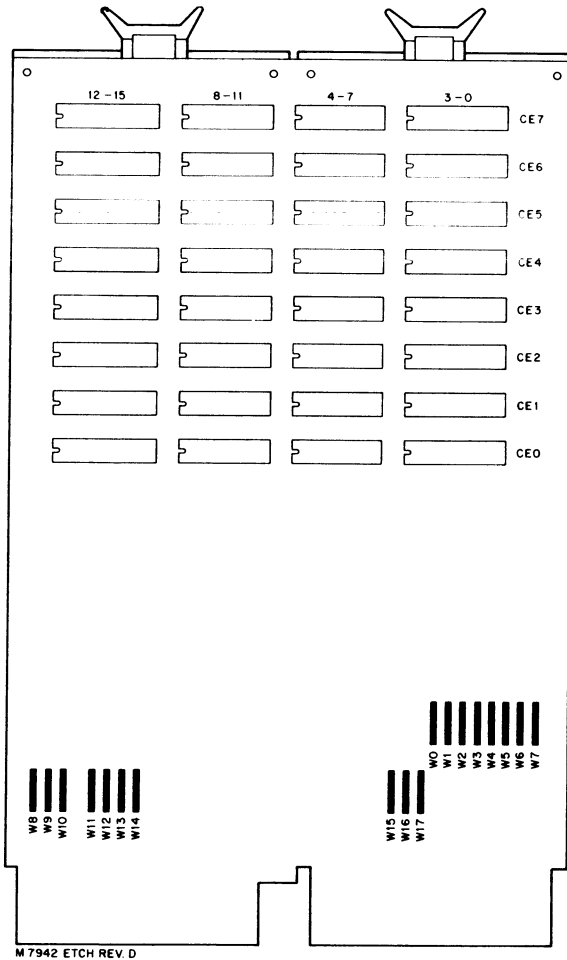


Figure 17-1 MRV11-AA Jumper Locations

Table 17-2 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

Table 17-2b 512 by 4-Bit PROM Addressing Within a Bank

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.

#### NOTE

Jumpers W8, W9, W10 are installed; W11, W12, W13, W14 are removed.

**Table 17-2c 256 by 4-Bit PROM addressing Within Lower 2K Portion of Bank**

<b>Reply Jumper</b>	<b>Physical Row</b>	<b>PROM Octal Address Range</b>
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

**NOTE**

Jumpers W11, W12, W13 are installed; W8, W9, W10, W14 are removed.

**Table 17-2d 256 by 4-Bit PROM Addressing Within Upper 2K Portion of Bank**

<b>Reply Jumper</b>	<b>Physical Row</b>	<b>PROM Octal Address Range</b>
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

**NOTE**

Jumpers W11, W12, W14 are installed; W8, W9, W10, W13 are removed.

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 17-2 illustrates addressing words used with the MRV11-AA. Refer to the addressing format for the type of PROMs or ROMs being used.

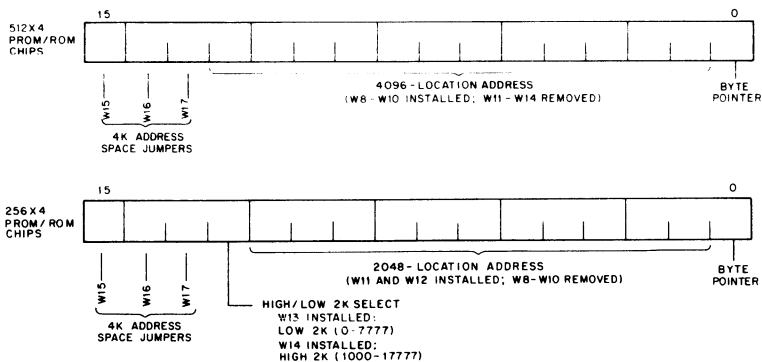


Figure 17-2 MRV11-AA Address Word Formats

### PROM Integrated Circuits

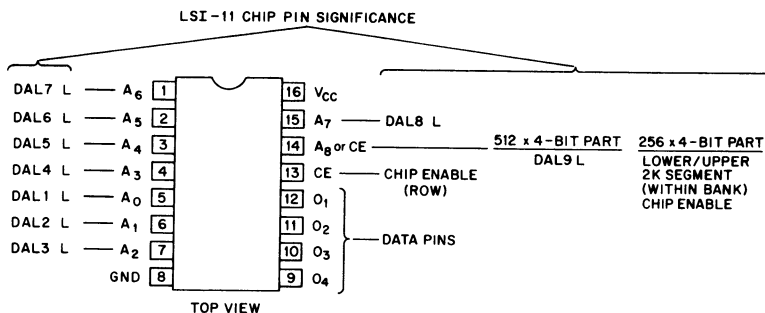
The actual procedure for loading data into PROMs (or writing specifications for masked ROMs) will vary, depending on the manufacturer. Those procedures are beyond the scope of this Handbook. (See PROM/ROM manufacturers' data sheets.) However, the user must be aware of the PROM pins versus LSI-11 data 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 handles four bits at a time. 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 17-3.

Table 17-3 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

PROM addressing is shown in Figure 17-3. 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 manufacturers' 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").

Figure 17-3 PROM/ROM Pin Addressing



## Programming PROMs

See the PROMs chapter, Chapter 18, for information about programming PROMs. Do not attempt to program PROMs until you are thoroughly familiar with the information contained in that chapter.

## DESCRIPTION

Major functions contained on the MRV11-AA module are shown in Figure 17-4. 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) constitute 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 DAL 13-15 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.

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 17-5. Bank select stored (SBS H) is gated to produce a low SEL L enable signal, which is applied to the D input of the

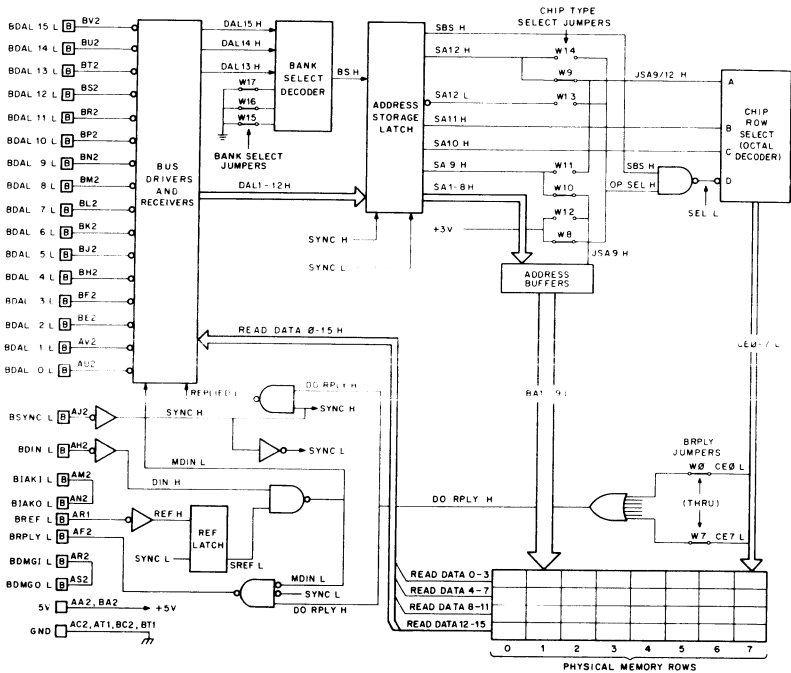


Figure 17-4 MRV11-AA Logic Block Diagram

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, jumper W8, W9, and W10 are removed and jumpers W11, W12, and either W13 or W14 are installed, as shown in Figure 17-6. SA10 and SA11 are applied to octal decoder A and B inputs, respectively. Bit SA9, which is not used to address the 256 by 4-bit chips directly, is then applied to input C of the octal decoder.

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.

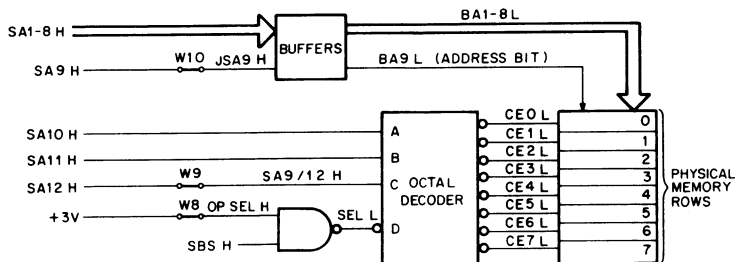


Figure 17-5 512 by 4-Bit Chip-Jumper Configuration

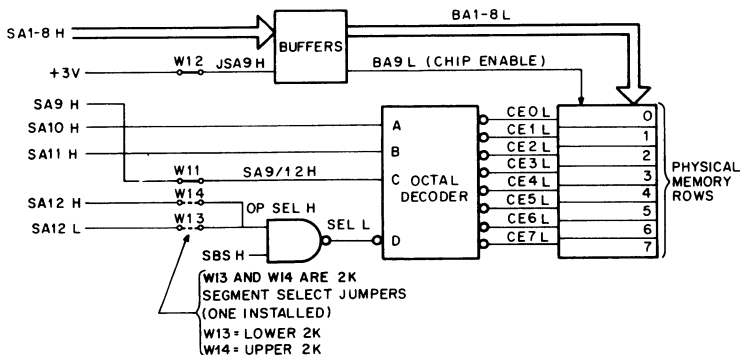


Figure 17-6 256 by 4-Bit Chip-Jumper Configuration

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 are 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 BDAL 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 are 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-AA 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 are 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****MRV11-BA LSI-11 UV PROM/RAM**

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 are lost. PROM contents are not volatile; programs and data stored in PROMs are available when operating power is restored.

Each MRV11-BA 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 — BENEFITS**

- 256K-word static on-board RAM— provides programs with read/write memory.
- Chip sockets — provide for installation of up to 4K words (8 KB) of PROM in 1K increments.
- Address jumpers — allow customized PROM and RAM memory configurations.
- Charge pump circuit — only the normal +5 Vdc and +12 Vdc operating voltages available from the LSI-11 Bus are required.

**SPECIFICATIONS**

Identification	M8021
Size	Double
Power	
LSI-11 UV PROM less	+5V $\pm$ 5% at 0.58 A
PROM integrated circuits	+12V $\pm$ 3% at 0.34 A
With eight 1K $\times$ 8	+5V $\pm$ 5% at 0.62 A
PROM integrated circuits	+12V $\pm$ 3% at 0.5 A

## Bus Loads

AC	2.8
DC	1.0

**CONFIGURATION**

Jumper locations are included on the MRV11-BA module as shown in Figure 17-7. 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.

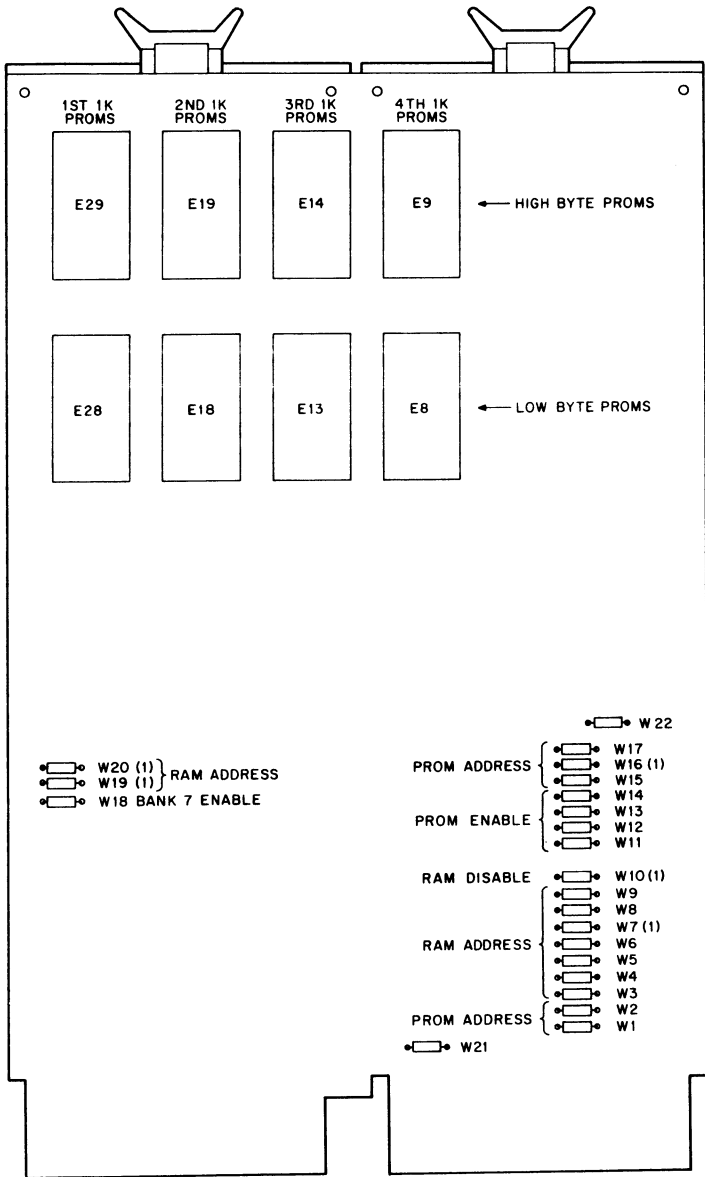
W21 and W22 control the MRV11-BA response to attempts to write in PROM locations. The module is factory-configured with W21 installed and W22 removed. When configured in this manner, any attempt to write in the PROM will result in a bus time-out error.

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:

<b>Mnemonic</b>	<b>Octal Code</b>	<b>Instruction</b>
MTPS	1064SS	Move byte to PS
MUL	070RSS	Multiply
DIV	071RSS	Divide
ASH	072RSS	Shift arithmetically
ASHC	073RSS	Arithmetic shift combined

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 the PROMs chapter 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



NOTE:  
(1) = JUMPERS NOT FACTORY INSTALLED.

Figure 17-7 MRV11-BA Jumper and Socket Locations

address. When this is done, RAM data will be accessed properly 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 17-8. Configure the RAM starting address by removing and/or installing the appropriate jumpers.

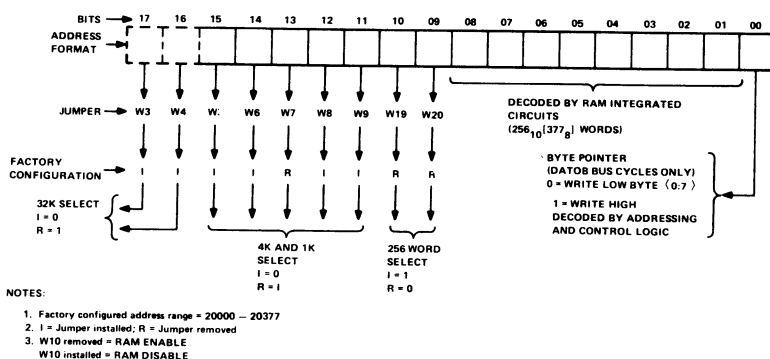


Figure 17-8 MRV11-BA RAM Addressing

### 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 17-9. Configure the PROM starting address by removing and/or installing the appropriate jumpers. Remove or install PROM size jumpers W11 through W14 as shown; 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.



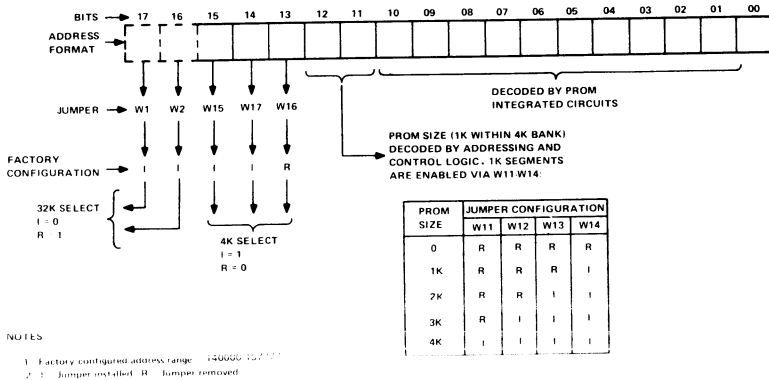


Figure 17-9 MRV11-BA PROM Addressing

### 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 damaged easily by static discharges because of 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 they are 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.

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 requires only 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 (+5V and +12V) 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.

**DESCRIPTION**

Four major functions constitute the MRV11-BA option: addressing and control logic,  $4K \times 16$  PROM array,  $256 \times 16$  RAM array, and charge pump circuit. These functions are shown in Figure 17-10. The PROM and RAM arrays constitute the actual memory portion of the module.

Four chip enable signals [CE<0:3> L] select the addressed pair of 1024-location by 8-bit 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 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.

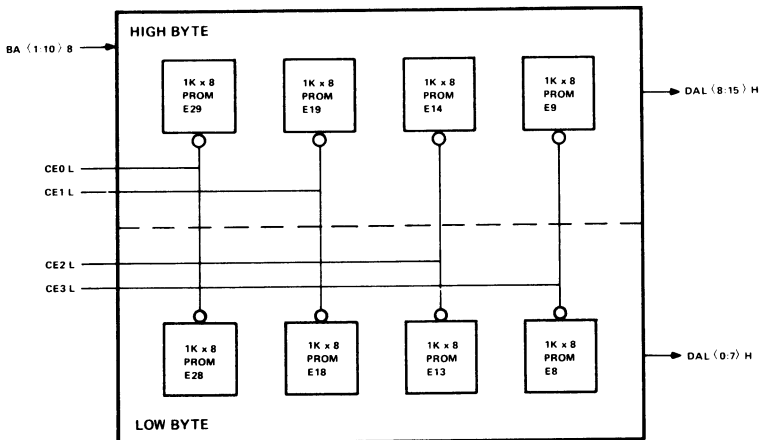


Figure 17-11 PROM Array

### RAM Array

Four factory-installed 256-location by 4-bit RAM integrated circuits constitute the RAM array, as shown in Figure 17-12. 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.

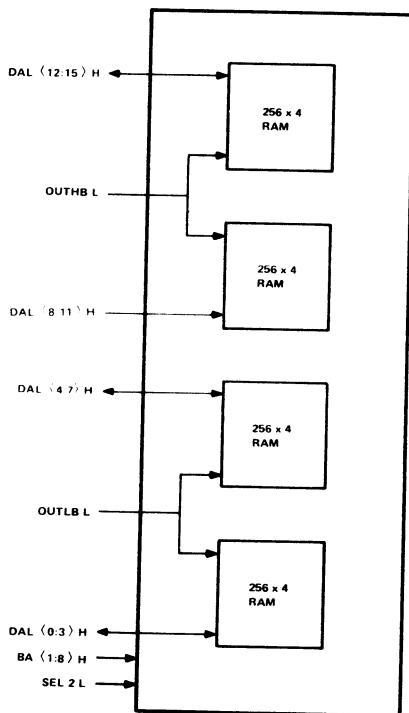


Figure 17-12 RAM Array

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 the OUT signal will go low, selecting the addressed byte.

### Addressing and Control Logic

Addressing and control logic functions are shown in Figure 17-13. Separate address decoding logic is included for PROM and RAM arrays. A common PROM/ROM address latch stores buffered address bits BA<1:8> 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.

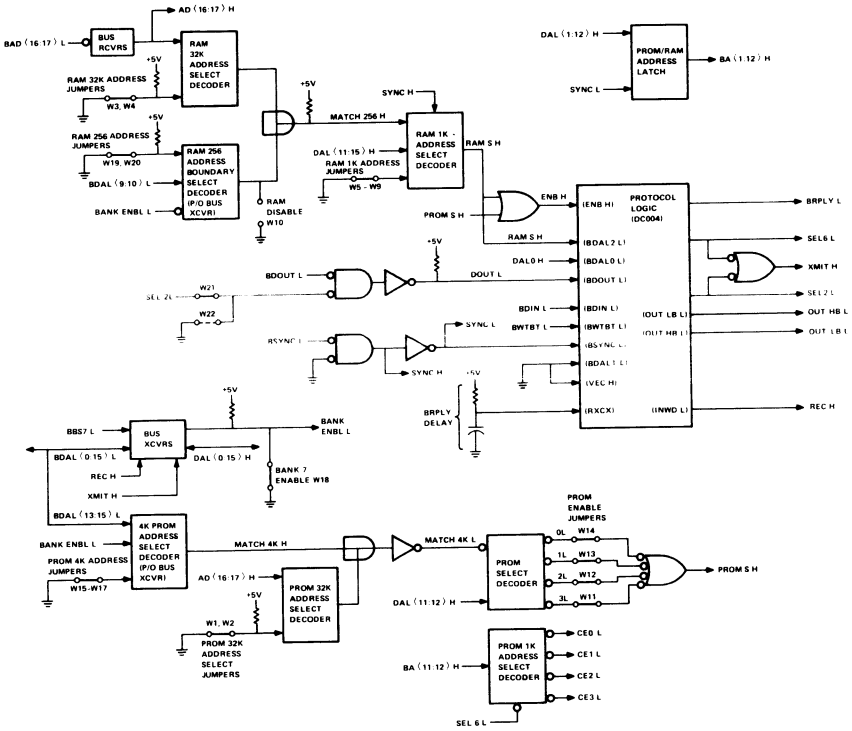


Figure 17-13 MRV11-BA Addressing and Control Logic

The addressing and control logic also includes bus transceivers that receive and transmit address and data bits to and from the LSI-11 Bus.

REC H, when high, inverts and gates BDAL<0:15> L bits onto the DAL<0:15> H lines. These lines form 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 on any 4K bank of system memory. PROM 32K address select decoder and jumpers W1 and W2 permit addressing in 128K memory 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 determines which PROM select decoder output will go active (low). Jumpers W11 through W14 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<1:12> H are applied to the PROM 1K address select decoder, producing one active chip enable signal (CE0 L through CE2 L) that enables the appropriate pair of 1K × 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 (BDAL2 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 ad-

dress 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 17-14). 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 are 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 are placed on BDAL<0:15> L where they 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 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.

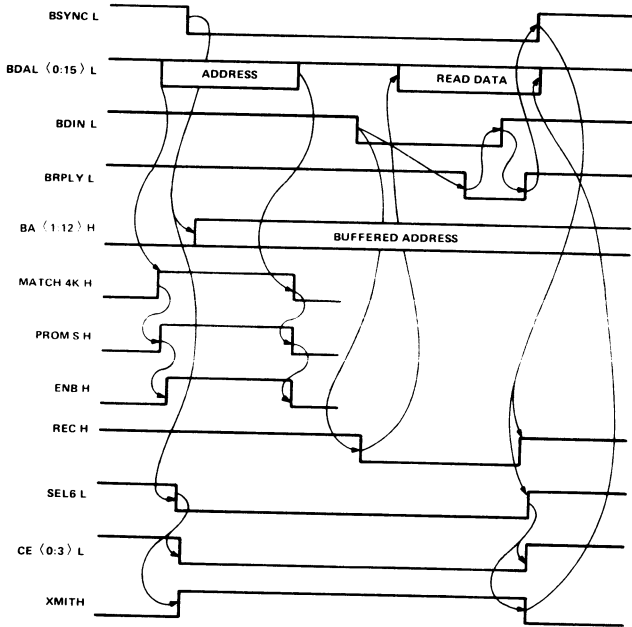


Figure 17-14 PROM Read Sequence (DATI)

**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 17-15) only when the 256 RAM is addressed (SEL2 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.

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.



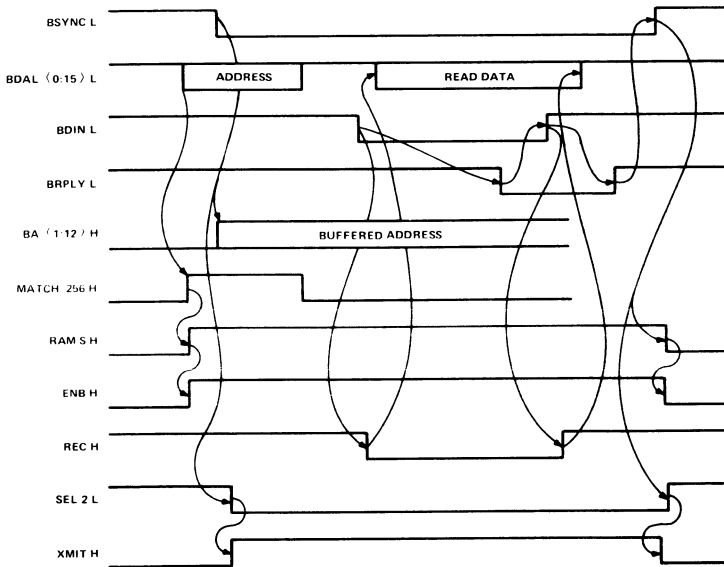


Figure 17-15 RAM Read Sequence (DAT1)

**RAM Read Sequence** A RAM read sequence is initiated when a bus master device places an address on the LSI-11 Bus (Figure 17-15). 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.

The active SEL2 L signal is applied to RAM integrated circuit chip enable inputs, enabling data to be read. Buffered address bits BA<11:8> 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 are 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 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 17-16), 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 are 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 not asserted (high) by the bus master (DATO bus cycle), or only the 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.

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.

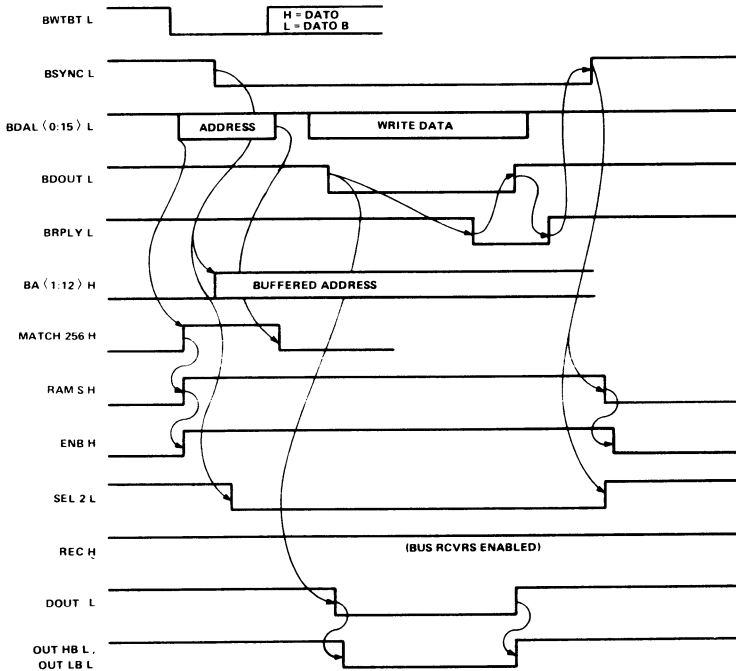


Figure 17-16 RAM Write Sequence (DATO or DATOB)

### Charge Pump Circuit

The charge pump circuit produces the  $-5\text{V}$  operating power for the PROM array integrated circuits. The basic components comprising the charge pump circuit are shown in Figure 17-17. Input power is obtained from the  $+12\text{V}$  present on the LSI-11 Bus. Hence, the MRV11-BA module does not require external power other than the usual  $+5\text{V}$  and  $+12\text{V}$  present on the backplane.

The oscillator provides the basic rectangular pulse that drives current switch Q3. When the oscillator turns Q3 on,  $+12\text{V}$  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 above), 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 10V. At this point, the zener voltage of D2 is exceeded and Q1 conducts. Q1 then produces a threshold control voltage that re-

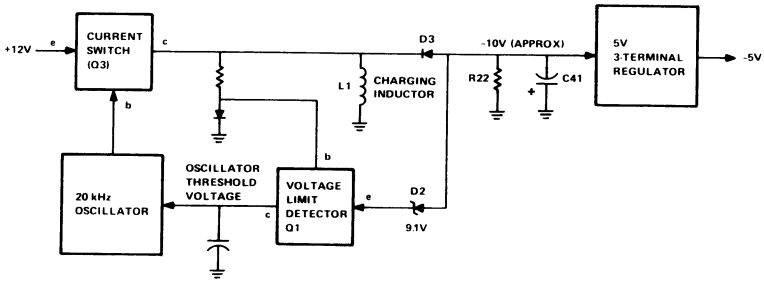


Figure 17-17 -5V Charge Pump Circuit (Simplified)

duces 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 -10V under any normal load conditions.

The actual regulated -5V output is produced by a 3-terminal, -5V regulator. The regulator also contains overcurrent and thermal overload protection circuits.

**MRV11-C****MRV11-C ROM MODULE**

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  $4K \times 8$  chips. Using these high-density chips gives the module a total capacity of 64 KB.

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 2 KB windows in memory address space to access 2 KB segments of the ROM array. The segments that are viewed through each window can be varied under program control. A bootstrap capability allows the top 256 words of any 2K word page to contain a bootstrap program.

**FEATURES — BENEFITS**

- 16K, 32K, or 64 KB of ROM — provide flexibility of memory size.
- Choice of EPROM, fusible link PROM, or masked ROM — provides flexibility of memory function.
- 18-bit addressing — allows the module to be configured in a system having up to 128K words of address space.
- Window-mapping — allows adjacent addresses to be located in non-adjacent physical address space.
- Bootstrap capability — no additional board is required to provide a bootstrap.

**SPECIFICATIONS**

Identification	M8048
Size	Double
Power	+5 Vdc, 0.8 A
Bus Loads	
AC	2.0
DC	1.0
ROM Specifications	
Power	+5V $\pm$ 5%
Pins	24 Pin Spacing
Access Time	up to 450 ns
Size	1K $\times$ 8, 2K $\times$ 8, or 4K $\times$ 8 bits

Type

See accompanying tables for a partial listing

**UV PROMs**

<b>UV PROMs</b>	<b>Chip Array Size</b>	<b>Max. Memory Size</b>
Intel 2758	1K × 8	16 KB
Intel 2716	2K × 8	32 KB
Intel 2732	4K × 8	64 KB
Mostek MK2716	2K × 8	32 KB
TI TMS 2516	2K × 8	32 KB
TI TMS 2532	4K × 8	64 KB

**PROMs**

<b>PROM</b>	<b>Chip Array Size</b>	<b>Max. Memory Size</b>
Intel 3628	1K × 8	16 KB
Signetics 82S 2708	1K × 8	16 KB
Signetics 82S 181	1K × 8	16 KB
Signetics 82S 191	2K × 8	32 KB

**CONFIGURATION**

The MRV11-C Read-Only Memory (ROM) contains 129 wire-wrap pins and 16 ROM chip sockets. The user configures module features by installing jumper wires between the wire-wrap pins. The user can configure the following items:

- Memory size
- Direct addressing mode
- Window mapping mode
- Bootstrap
- Use of multiple MRV11 boards
- ROM chips
- Chip access time
- DATIO bus cycle inhibit

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. All the ROM chips must be the same array size; that is, either 1K by 8, 2K by 8, or 4K by 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: 16 KB, 32 KB, or 64 KB. Subsets of these sizes can also be chosen as shown in Table 17-4. In addition, the user can configure the MRV11-C to be part of a system with more than one MRV11-C module.

**Table 17-4 Storage Capacity per Board as a Function of Chip Array Size and Number of Chips**

Number of Chips Installed	Chip Array Size		
	2758 (Typ.) 1024 × 8	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

The MRV11-C ROM module operates in either the direct addressing mode or the window mapping mode. In the direct addressing mode, the user's program addresses physical memory directly. In the window mapping mode, the user's program addresses a continuous virtual address space which the MRV11-C breaks up into 2K segments of physical address space. The 2K segments may not be next to each other in physical memory.

The user can also select the starting address of a bootstrap on the module (within the bootstrap address region: 17300<sub>8</sub> to 173776<sub>8</sub>). The bootstrap option can be used in either mode or disabled.

The chip access time must be selected to accommodate the chips with the slowest access time. The user should also inhibit DATIO bus cycles to prevent attempted writes to read-only memory.

To configure this module, the user should follow the flowchart shown in Figure 17-18. The physical locations of the pins are detailed in Figure 17-19. The module is shipped from the factory with no jumper wires installed.

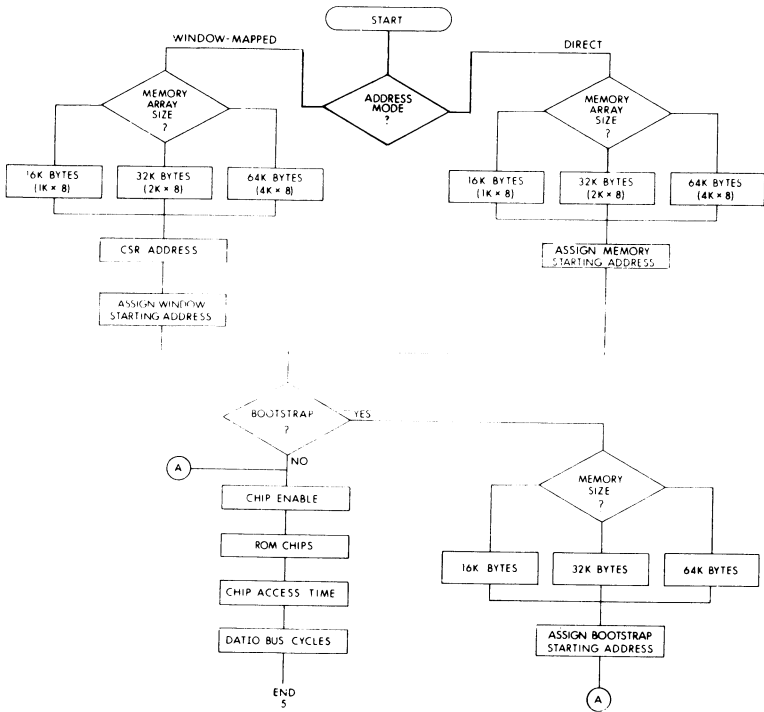


Figure 17-18 Configuration Guide

### Direct Addressing Mode

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 8 KB boundary from 0 to 248 KB (addresses 00000<sub>8</sub> to 76000<sub>8</sub>). When operated in this mode, the application program executes directly from the MRV11-C physical memory.

In direct mode, address bits AD11 through AD15 are used to access data in the ROM. Bits AD14 and AD15 are 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 64 KB systems, but all five address bits (AD13 through AD17) are used for 256 KB systems. The starting address must be configured to start on 8 KB boundaries as determined by the user.



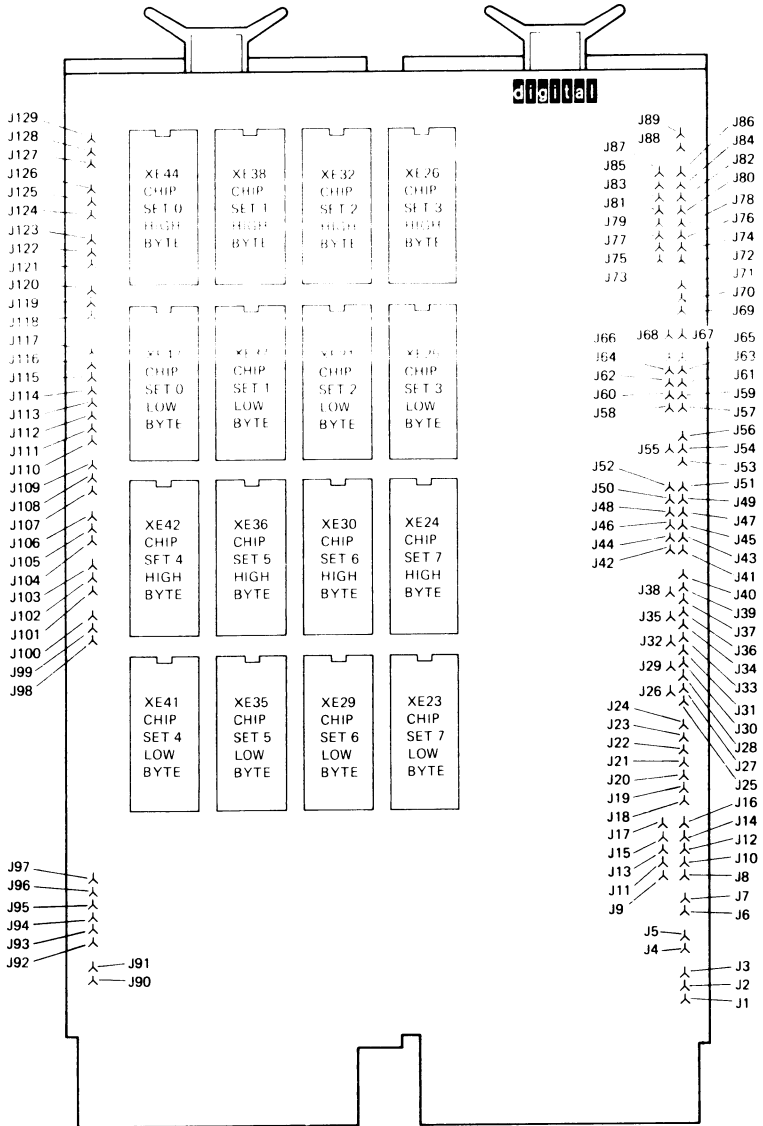


Figure 17-19 MRV11-C Wire-wrap Pin Locations

The range of the direct addresses required depends on the amount of memory installed on the module. The minimum is 2 KB and the maximum is 64 KB. Once the address space is determined, the starting address is configured by installing jumper wires. All the jumper wire configurations for the 8 KB (4K word) boundaries are listed in Table 17-5.

The starting address and the bank of addresses assigned determine the addressing sequence of the ROM chips. Figure 17-20 shows examples of 32 KB and 64 KB 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 in correct sequential order.

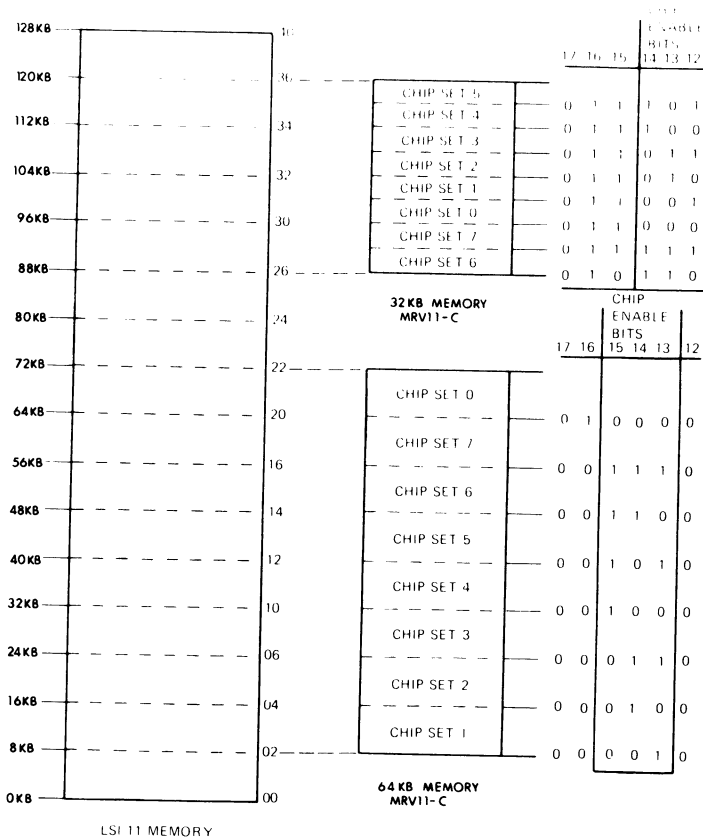


Figure 17-20 Typical MRV11-C Memory Mapping

Table 17-5 Jumper Wire Configurations for 8 KB Boundaries

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

For a 64 KB memory the desired starting address is  $20000_8$ , at the 8 KB boundary. For this size memory, address bits 13, 14, and 15 are decoded to select one of the eight pairs of ROM sockets. Expanding address  $20000_8$  in binary (i.e.  $000\ 010\ 000\ 000\ 000\ 000$ ) shows that with this starting address, the first chip set selected is chip set 1.

The starting chip set can be determined similarly for a 32 KB memory. In this case, address bits 12, 13, and 14 are the chip enable bits, with a starting address at the 88 KB boundary (address  $260000_8$ , or  $010\ 110\ 000\ 000\ 000\ 000$ ). Chip set 6 is the first set to be accessed. Likewise, a 16 KB memory uses address bit 11, 12, and 13 as the chip select bits.

Tables 17-6, 17-7, and 17-8 respectively summarize the proper jumper locations for 16 KB, 32 KB, and 64 KB direct addressing modes.

**Table 17-6 16 KB Direct Addressing Jumpers**

<b>Function</b>	<b>Jumpers Installed</b>
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

**Table 17-7 32 KB Direct Addressing Jumpers**

<b>Function</b>	<b>Jumpers Installed</b>
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

**Table 17-8 64 KB Direct Addressing Jumpers**

<b>Function</b>	<b>Jumpers Installed</b>
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

**Window Mapping Mode**

When window mapping mode is selected, the entire ROM is not visible to the LSI-11 address space at any particular point in time. Instead, any two 2 KB segments of the ROM can be addressed through two independent windows defined by the system's address space. The association of segments of the ROM board with windows is controlled by a Control and Status Register (CSR).

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.

**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<sub>8</sub> to 770000<sub>8</sub>. To maximize the amount of space left for system RAM, a default window base of 160000<sub>8</sub> (760000<sub>8</sub> for LSI-11/23 systems) is suggested.

Each MRV11-C uses one 16-bit CSR located in the system I/O page to determine mapping of ROM segments into windows. The default address for this CSR is 177000<sub>8</sub> (777000<sub>8</sub> in the LSI-11/23 system). The valid address range for CSRs is 177000<sub>8</sub> to 177036<sub>8</sub> (777000<sub>8</sub> to 777036<sub>8</sub> on LSI-11/23s). Figure 17-21 shows the bit assignments for the MRV11-C control and status register. Table 17-9 lists the control and status register addresses.

Table 17-9 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	
177004	R	R		R
177006	R	R		
177010	R		R	R
177012	R		R	
177014	R			R
177016	R			
177020		R	R	R
177022		R	R	
177024		R		R
177026		R		
177030			R	R
177032			R	
177034				R
177036				

R = Jumper Removed

I = Jumper Installed

**NOTE**

Install J67 to J68 to allow the use of bit 15 of the CSR

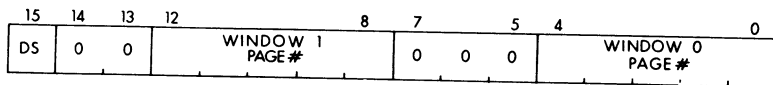


Figure 17-21 MRV11-C Control and Status Register Format

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 2 KB 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 read-only and will always 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.

Tables 17-10, 17-11, and 17-12 respectively summarize the proper jumper connections for 16 KB, 32 KB, and 64 KB window mapping modes.

**Table 17-10 16 KB Window Mode Jumpers**

<b>CSR Output</b>	<b>Jumpers Installed</b>
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

**Table 17-11 32 KB Window Mode Jumpers**

<b>CSR Output</b>	<b>Jumpers Installed</b>
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

**Table 17-12 64 KB Window Mode Jumpers**

<b>CSR Output</b>	<b>Jumpers Installed</b>
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

**Starting Address of Windows** — Wire-wrap pins J41 through J52 are used to configure the starting address of windows. The user selects an address and installs the jumper wires as directed in Figure 17-22. The recommended value of the window starting is  $160000_8$  (or  $760000_8$  for 18-bit systems). This places the window at the bottom of the I/O page.



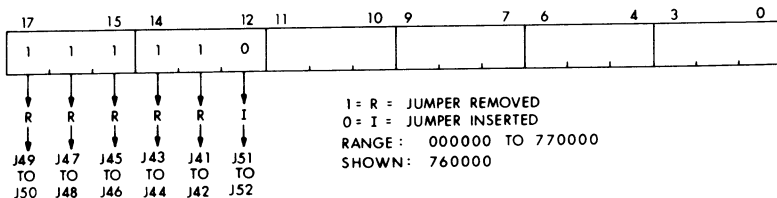


Figure 17-22 Selecting Window Starting Address

**NOTE**

The MRV11-C does not select the I/O page on the BBS7 (bank select 7) signal for windows placed in the I/O page. Therefore, the entire address for the window must be asserted by the processor and decoded by the MRV11-C. An 11/23 processor will only assert an address of  $760000_8$  in 18-bit mode (i.e., when the memory management unit is enabled) and bank 7 is mapped to the I/O page.

**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  $173000_8$  and for 18-bit systems at  $773000_8$ . The bootstrap program is normally enabled and must be disabled if it is not being used. To disable the bootstrap, install a jumper wire between wire-wrap pins J88 and J89. The bootstrap program is inserted as the top 512 bytes of any 2 KB page of ROM. The user installs jumper wires for the boot multiplexer 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.

Refer to Tables 17-13, 17-14, and 17-15 respectively to jumper the bootstrap starting address of 16 KB, 32 KB, and 64 KB ROM memory systems.

**Use of Multiple MRV11-C 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  $177000_8$  ( $777000_8$  in LSI-11/23 systems). Refer to Table 17-9 to configure CSR addresses. 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.

**Table 17-13 Bootstrap Starting Address: Jumper Configurations for 16 KB ROM Memory**

Starting Address	Install Jumper Wire From		
	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

Logic 1 = J23

Logic 0 = J24

Bootstrap starting address is normalized to memory location 000000.

**Table 17-14 Bootstrap Starting Address: Jumper Configurations for 32 KB ROM Memory**

Starting Address	Install Jumper Wire From			
	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

Logic 1 = J23

Logic 0 = J24

Bootstrap starting address is normalized to memory location 000000.

**Table 17-15 Bootstrap Starting Address: Jumper Configurations for 64 KB 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

Logic 1 = J23

Logic 0 = J24

The window enable bit of the CSR (bit 15) is used to provide the user software control over the windows. Setting bit 15 to 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 module will be disabled on power-up.

When enable/disable is implemented, the disabled 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.

### ROM Chips

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 accidentally 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 Table 17-16 below.

**Table 17-16 Chip-Enable Jumpers**

<b>Sockets Enabled</b>	<b>Chip-Enable Signal</b>	<b>Wire-wrap Pins Jumpered</b>
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

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  $1K \times 8$  bit,  $2K \times 8$  bit, or  $4K \times 8$  bit ROMs. When the MRV11-C is fully populated, the result will be either 16K, 32K, or 64 KB 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 wire-wrap pins for the user to select A11, A12, 5 Vdc or ground. There are three individual loops that interconnect all chips and three wire-wrap pins available for each individual chip. Wire-wrap 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. Wire-wrap pin J114 interconnects wire-wrap pins that are individually associated with each chip. Pin 18 of each chip is individually wired to a wire-wrap pin and chip pin 20 is wired to the chip-enable signal. Chip pin 20 is also individually wired to a wire-wrap 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 Figure 17-23 below, there are pin configurations for two types of chips, a  $2K \times 8$  ROM that is used for 32 KB memories and a  $4K \times 8$  ROM that is used for 64 KB memories. To configure the 32 KB 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 32 KB 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  $4K \times 8$  ROM to configure a 64 KB 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/ $V_{PP}$  (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 64 KB 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

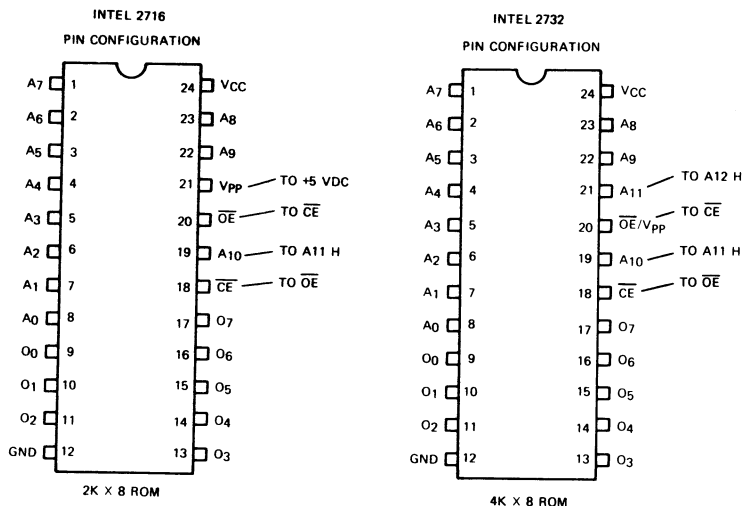


Figure 17-23 Pin Configuration

**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 wire-wrap 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 wire-wrap pins J1 and J3 is removed and the jumper wire between wire-wrap pins J2 and J3 remains inserted.

**DATIO Bus Cycle Inhibit**

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). This condition is allowed unless a jumper wire is installed between wire-wrap pins J4 and J5. With this jumper installed, the BDOOUT 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.

**Wire-wrap Pin Identification**

The MRV11-C module provides the user with 129 wire-wrap pins to configure the module for many types of applications. These wire-wrap pins are identified and located on the module in Figure 17-19. In Table 17-17 below, the wire-wrap pins are numerically listed with descriptions of their functional use.

**Table 17-17 Wire-wrap Pin Identification**

<b>Wire-wrap Pin Designation</b>	<b>Function</b>
J1	RXCX pull-up resistor
J2	RXCX optional capacitor
J3	RXCX signal
J4	LMATCH input for BDOOUT control
J5	LMATCH for BDOOUT control

<b>Wire-wrap Pin Designation</b>	<b>Function</b>
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



<b>Wire-wrap Pin Designation</b>	<b>Function</b>
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

<b>Wire-wrap Pin Designation</b>	<b>Function</b>
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

<b>Wire-wrap Pin Designation</b>	<b>Function</b>
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 wire-wrap 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 wire-wrap 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 wire-wrap 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 wire-wrap 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 wire-wrap 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

<b>Wire-wrap Pin Designation</b>	<b>Function</b>
J118	Pin 18 input for chip set 0
J119	Chip wire-wrap 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 wire-wrap 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 wire-wrap 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 wire-wrap interconnection for chip set 3
J129	Pin 20 input for chip set 3 (Chip Enable 3)

## **PROGRAMMING**

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 56 KB 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 in Figure 17-24. 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 <<ADRS/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
MFPS4	(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

Figure 17-24 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

MRV11-C 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. Figure 17-25 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 pro-
                                ;gram?
          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

```

Figure 17-25 Bootstrap Loader for Stand-Alone Programs in RT-11  
.SAV Format

## DESCRIPTION

The MRV11-C ROM module operates in either the direct addressing mode or the window mapping mode. The configuration of these modes and the optimal bootstrap area is controlled in the module by three multiplexers: the CSR low byte and direct address MUX, the CSR high byte MUX, and the boot MUX. Refer to Figure 17-27.

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 with the ROM. There are three multiplexers that provide inputs to the decoder. The minimum input is three bits for a 16 KB 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.

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 is 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 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.

CSR address information received from the bus is decoded by four DC005 transceiver chips in the bus address interface. 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 take data to or from the bus. The CSR is the only location that receives data from the bus. The ROM will always transfer data to the bus.

The I/O control monitors 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 CSR. 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 (pin AN1) is used by the bus to remotely disable the MRV11-C module. Note that this signal is not bussed in DIGITAL backplanes, but may be wired in backplanes provided by other manufacturers.

When the bootstrap is used and the bus address is in the bootstrap region (173000<sub>8</sub>-173776<sub>8</sub>) 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.





**PROM**

## CHAPTER 18

### PROMS

This chapter contains specific instructions for programming, loading, and erasing PROMs for the MRV11-AA, MRV11-BA, MRV11-C, and MXV11-A. Also included is a description of two options for blasting PROMs. PB11 is a software/hardware option for programming PROMs. It consists of a universal PROM blaster (RS232-C) and a software utility which runs under RT-11 (V03B or later). The other option is QJV11, a PROM-formatting program that accepts binary object program paper tapes and produces PROM listings and paper tapes for use with an automatic PROM programmer.

#### PROGRAMMING NOTES

Generally, programs or data that can be read from read/write memory can also be executed from a PROM. However, certain instructions, when executed on the LSI-11 and the LSI-11/2, fetch the source operand via a DATIO bus cycle (read/modify/write) rather than a DATI bus cycle. When using an MRV11-AA, this will cause a bus time-out error. The other PROM modules have jumpers which allow them to respond to DATIO without timing out. The following instructions do a DATIO cycle:

Mnemonic	Octal Code	Function
CLR	0050DD	Clear destination
CLRB	1050DD	Clear destination (byte)
MOVB	11SSDD	Move source to destination (byte)
MTPS	1064SS	Move byte to PS
MUL	070RSS	Multiply
DIV	071RSS	Divide
ASH	072RSS	Arithmetic shift
ASHC	073RSS	Arithmetic shift combined

Precautions must be taken if a module that is configured to reply to DATIO cycles is used in a system running DIGITAL software or bootstrap. Most DIGITAL software, such as the RT-11 operating system, determines memory size by attempting to write into a location in memory. If a time-out occurs, it is ensured that no memory is present at that location. PROM memory will normally time-out, as it does not respond to a write cycle. When the PROM module is configured to reply to write cycles, DIGITAL software will assume that 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.

To avoid this potential problem, include separate MOV instructions within the program. First, move the source operand from the PROM location to a general-purpose register or a location in read/write memory. Then execute the instruction using the general-purpose register or read/write memory location as the source operand.

Two examples are shown below using general-purpose register R4 and memory location TEMP as the source operand.

Using a general-purpose register:

```
MOV NEWPS,R4      ;MOVE SOURCE OPERAND FROM PROM
                  ;TO TEMPORARY
                  ;(GENERAL PURPOSE) REGISTER

MTPS R4           ;MOVE NEWPS TO PS
```

Using a 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
```

## USING PROMS

Loading (blasting, burning, or programming) PROMs is the process by which the binary information is stored in the PROM location. This is a process that must be carefully executed as directed by the appropriate PROM loader manufacturer's instructions.

The following procedures describe loading and installing information for the MRV11-AA, MRV11-BA, MRV11-C, and MXV11-A.

### MRV11-AA Procedures

**PROM Types** — Basically, two general types of PROMs can be used in the MRV11-AA module:  $512 \times 4$  bit and  $256 \times 4$  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 constitute either a  $256 \times 16$  or  $512 \times 16$  read-only function. Recommended types are listed in Table 18-1.

Table 18-1 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

\* These parts are identical

When programming MRV11-AA PROMs for use as an RT-11 bootstrap for systems of 64 KB or less, use 256 × 4 PROMs. This will allow the addresses to be configured in the 173000-173776 range. Processor module power-up mode 2 can then be used for automatically bootstrapping RT-11 during system power-up. 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.

**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  $12134_8$  or  $101000101110001_2$ . The binary bits are stored in PROMs numbered from 1 to 4. Output pins, as indicated, will yield the read data bits for this instruction when addressed. The MRV11-AA data format is illustrated in Figure 18-1.

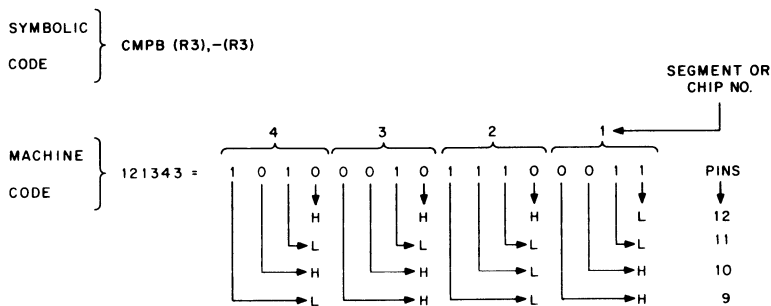


Figure 18-1 MRV11-AA Data Format

Since the word format is contained in four 4-bit slices (one slice in each PROM), 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 the 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 Table 18-2. Note that  $256 \times 4$  bit and  $512 \times 4$  bit PROMs are addressed in exactly the same manner, except for pin 14, which is A8 in the  $512 \times 4$  bit part, and CE in the  $256 \times 4$  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.

The MRV11-AA address format for  $512 \times 4$  and  $256 \times 4$  PROM applications is shown in Figure 18-2. 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 ( $256 \times 4$  PROM applications only) are jumper-configured on the MRV11-AA.

**Installing PROMs** — After the PROMs are properly programmed, loaded, and verified, they can be installed on the properly configured MRV11-AA module as illustrated in Figure 18-3. Observe that PROMs are installed in sets of four—one for each segment. Segment and set numbers correspond to those indicated in the QJV11 listing output.

An addressing summary from PROM sets as arranged by physical locations (CE numbers marked on the MRV11-AA module) is provided in Table 18-3.

Table 18-2 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	00000000	H	H	H	H	H	H	H	H	
2	00000010	H	H	H	H	H	H	H	L	
4	00000100	H	H	H	H	H	H	L	H	
6	00000110	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	L	H	
14	000001100	H	H	H	H	H	L	L	L	
16	000001110	H	H	H	H	H	L	L	L	
20	000010000	H	H	H	H	L	H	H	H	
⋮										Actual Logic Levels Required (256 <sub>10</sub> Locations)
774	111111100	L	L	L	L	L	L	L	H	
776	111111110	L	L	L	L	L	L	L	L	

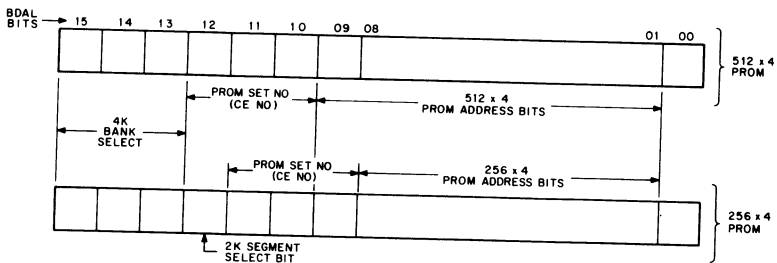


Figure 18-2 MRV11-AA Address Word Format

Table 18-3 MRV11-AA PROM Addressing Summary

Set No.	512 × 4 PROMs			256 × 4 PROMs		
	Address Range		Physical Location	Address Range		Physical Location
	Decimal	Octal		Decimal	Octal	
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

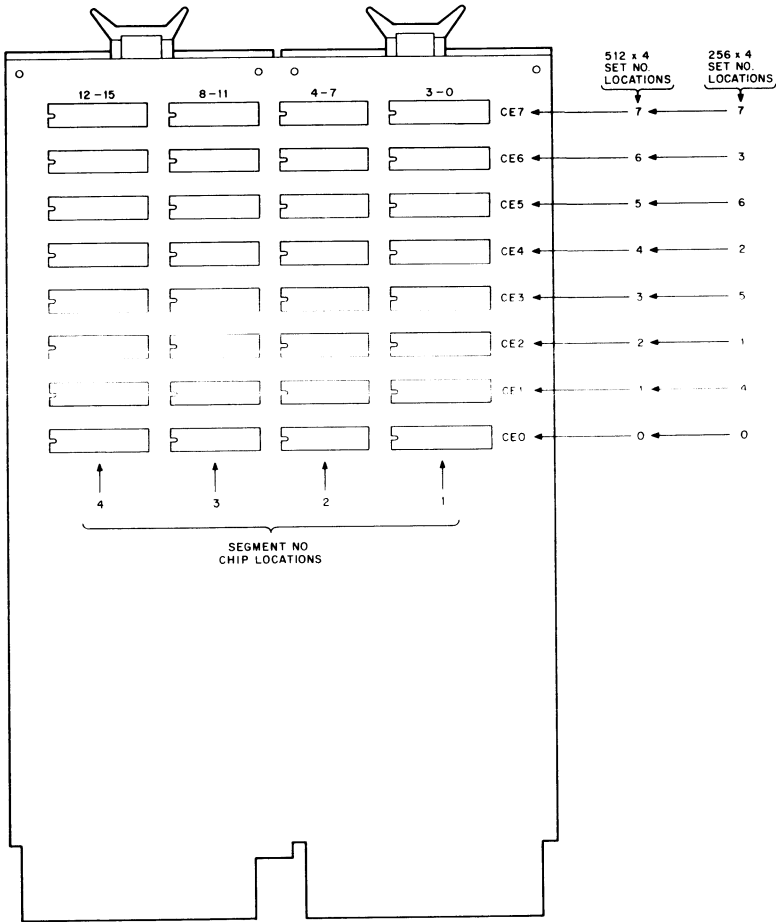


Figure 18-3 PROM Set and Segment Positions on the MRV11-AA Module

### MRV11-BA Procedures

**PROM Types** — The MRV11-BA contains eight sockets in which UV-erasable PROMs may be installed. Each socket will take one 1K x 8 bit EPROM (MRV11-BC, Intel 2708, or equivalent) for a total of 4K words of PROM memory.

**Data Word Format** — Each PROM word, when read by the processor, is stored in two bytes in two separate PROMs. Each word is simulta-

neously 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 PB11 or QJV11 PROM formatting utilities described later.

**Addressing** — PROM integrated circuits, when installed in the MRV11-BA module, are addressed by the 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 Table 18-4. 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 the ROM Memories chapter, Chapter 17. 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 A2537. The recommended integrated light (light intensity  $\times$  exposure time) is 10 W-s/m<sup>2</sup>. 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

**PROM Types** — The MRV11-C has 16 sockets which accept a variety of masked PROMs, fusible link PROMs, and UV erasable PROMs for a total capacity of 64 KB of memory. The MRV11-C accepts 24-pin, 5 volt only parts. The following PROMs (or equivalent) may be used:

UV PROMs	Chip Size	Max. Memory Capacity per MRV11-C
Intel 2758	1K $\times$ 8	16 KB
Intel 2716	2K $\times$ 8	32 KB
Intel 2732	4K $\times$ 8	64 KB
Mostek MM2716	2K $\times$ 8	32 KB
TI TMS 2516	2K $\times$ 8	32 KB



Table 18-4 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	L
2	0000000001	L	L	L	L	L	L	L	L	L	H	H
4	0000000010	L	L	L	L	L	L	L	L	H	L	L
6	0000000011	L	L	L	L	L	L	L	H	H	L	L
10	0000000100	L	L	L	L	L	L	L	H	L	L	L
12	0000000101	L	L	L	L	L	L	L	H	L	L	L
14	0000000110	L	L	L	L	L	L	L	H	H	L	L
.	.	.	.	.	.	.	.	.	.	.	.	.
3776	1111111111	H	H	H	H	H	H	H	H	H	H	H

} Actual Logic Levels Required (1024<sub>10</sub> Locations)

\*Bus address bit 0 is not used; hence, only even-numbered addresses are shown.

<b>UV PROMs</b>	<b>Chip Size</b>	<b>Max. Memory Capacity per MRV11-C</b>
TI TMS 2532	4K × 8	64 KB

<b>Bipolar PROMs</b>	<b>Chip Size</b>	<b>Max. Memory Capacity per MRV11-C</b>
Intel 3628	1K × 8	16 KB
Signetics 82S2708	1K × 8	16 KB
Signetics 82S181	1K × 8	16 KB
Signetics 82S191	2K × 8	32 KB

**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 in the high state.

**Addressing** — The MRV11-C uses non-inverted addressing. The address bits are asserted in the high state.

**Installing PROMs** — PROMs are installed in pairs in their appropriate sockets. See the ROM Memories Chapter.

### **MXV11-AA,AC Procedures**

**Types of PROMs** — The MXV11-AA,AC has two sockets available for user or bootstrap PROMs. The following types of PROMs (24-pin, 5 volt only parts, or equivalent parts) may be used:

<b>UV PROMs</b>	<b>Chip Size</b>	<b>Memory Size</b>
Intel 2758	1K × 8	2 KB
Intel 2716	2K × 8	4 KB
Intel 2732	4K × 8	8 KB
Mostek MK2716	2K × 8	4 KB
TI TMS 2516	2K × 8	4 KB
TI TMS 2532	4K × 8	8 KB

<b>Bipolar PROMs</b>	<b>Chip Size</b>	<b>Memory Size</b>
Intel 3628	1K × 8	2 KB

Bipolar PROMs	Chip Size	Memory Size
Signetics 82S2708	1K × 8	2 KB
Signetics 82S181	1K × 8	2 KB
Signetics 82S191	2K × 8	4 KB

**Data Word Format and Addressing** — The PROMs may be addressed to respond to Bank 0, Bank 1, or the bootstrap address (173000 or 773000). Please refer to the ROM Memories Chapter for detailed information on socket pinout and configuration.

### **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 paper tape 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 customer-installed/customer-supported 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 — BENEFITS

- Desk-top size — convenient to use.
- Optional adapters for different PROMs — wide variety of applications.
- Supported under RT-11 development software — no special programming required.
- Supports application development under PROMmable FORTRAN and MicroPower/Pascal — user need not know Macro assembly.
- Includes on-line diagnostic for the PROM programmer — increases system reliability.
- Connects to serial line interface on host via RS-232C — provides convenient communication between host and PROM programmer.

## 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 45°C (-40°F to 131°F)
Humidity	0 to 90% (non-condensing)

### Electrical

Input Power Range	100, 120, 220, 240 Vac (internally selectable), 50-60 Hz $\pm$ 10% at 35 Watts, UL listed and CSA certified
-------------------	---

## Model Designations

PB11-AY	Desk-top universal PROM programmer with a 25 foot (7.6 m) EIA cable for connection to RT-11 (V03B or later) system (PDP-11/03 or PDP-11/34) using a serial interface (DLV11-E, DLV11-F, DLV11-J or DL11-W and DL11-E). Includes customer-installed/customer-supported software on RX01 disk. Adapter kit and RS-232C cable must be purchased with the unit.
---------	---

PB11-AQ	Same as PB11-AY, includes customer installed/customer-supported 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.

### Hardware Requirements

The PROM programmer comes with a 25 foot (7.6 m) EIA cable for connection to a disk-based PDP-11/03 or PDP-11/34 system operating under RT-11 (V03B 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.

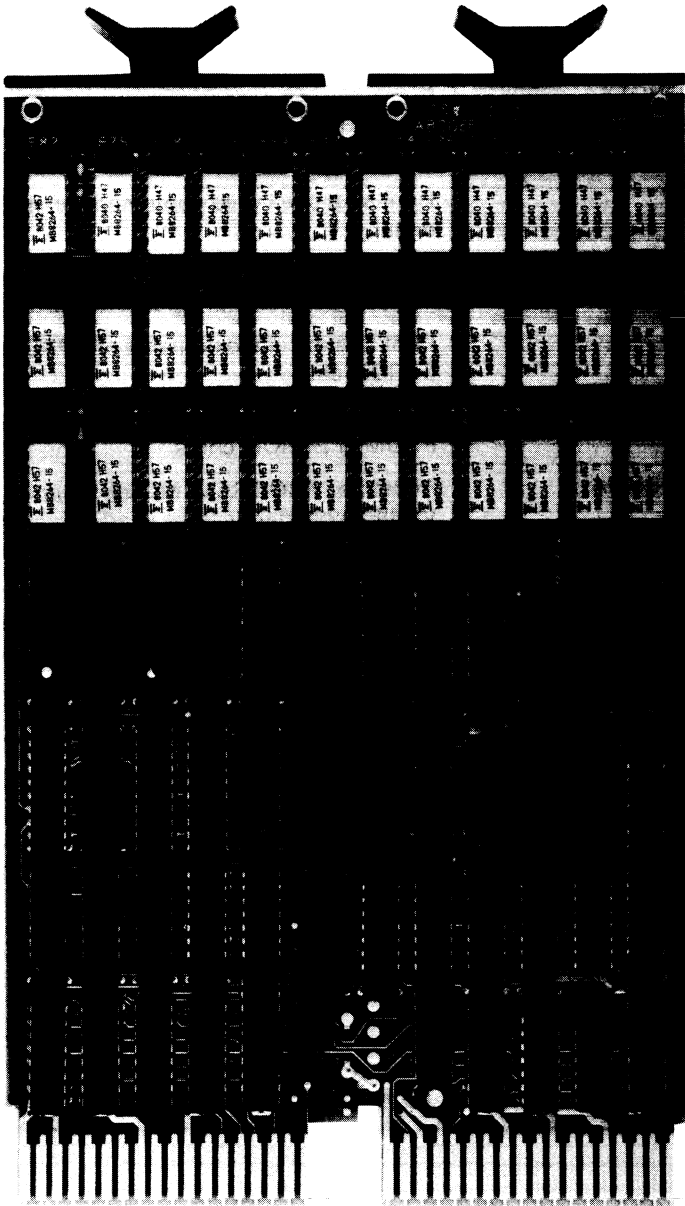
RT-11 is a powerful foreground/background operating system that supports DIGITAL's special PROMmable FORTRAN—a FORTRAN package uniquely designed for PROM applications.

The PROMmable FORTRAN (RT-11 FORTRAN IV V2.5) 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 run time 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 PROMs 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:

<b>Command</b>	<b>Function</b>
COPY	Copy 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	Redefine 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



## CHAPTER 19

# RAM MEMORIES

### **MSV11-B 4K BY 16-BIT MOS READ/WRITE MEMORY**

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 — BENEFITS**

- 4096 by 16-bit word — fewer boards required for full memory.
- 550 ns access time — data available quickly.
- Lower power — needs less cooling and smaller power supplies.
- User-configured 4K addresses — allow easy memory layout.

### **SPECIFICATIONS**

Identification	M7944
Size	Double
Power	+5 V $\pm$ 5% at 0.6 A +12 V $\pm$ 3% at 0.54 A
Bus Loads	
AC	1.9
DC	1.0

### **CONFIGURATION**

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**

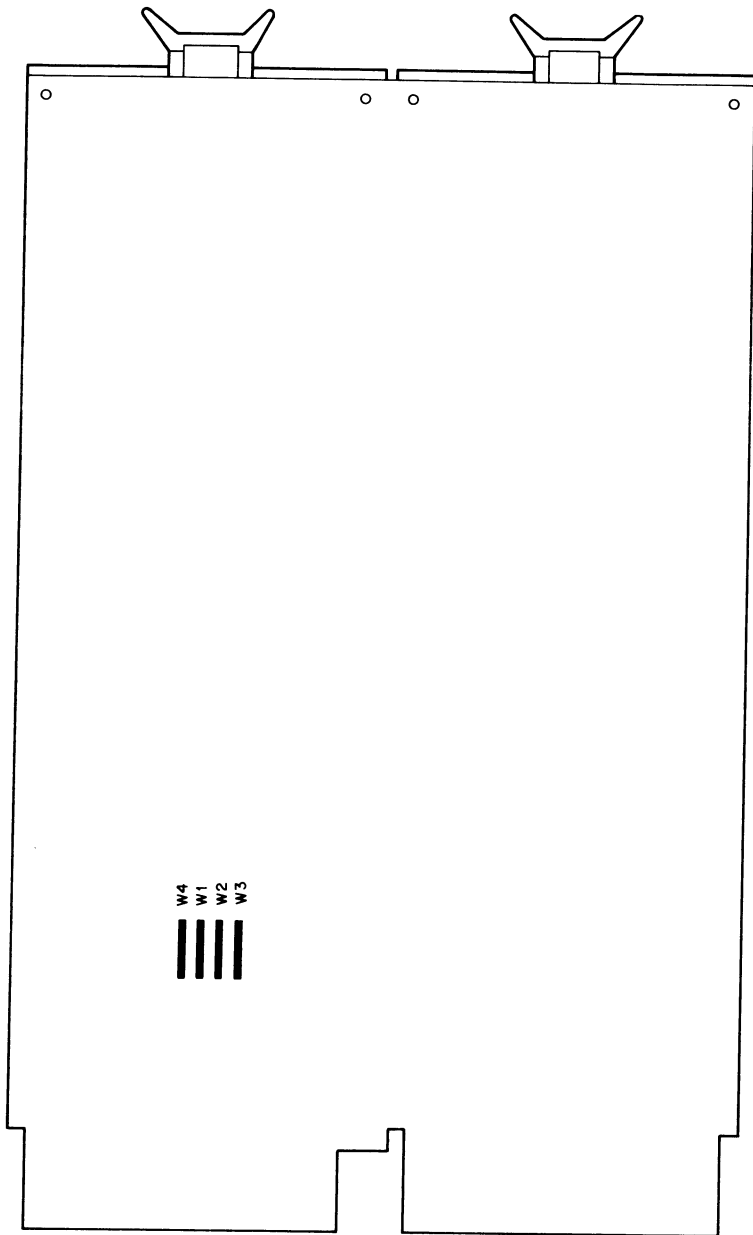
MSV11-B address jumpers are located as shown in Figure 19-1. The module is supplied with all address jumpers installed. Figure 19-2 illustrates a 16-bit address and how jumpers are assigned for the MSV11-B module.

**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 be the farthest physically away 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.



M7944 ETCH REV B

Figure 19-1 MSV11-B Jumper Locations

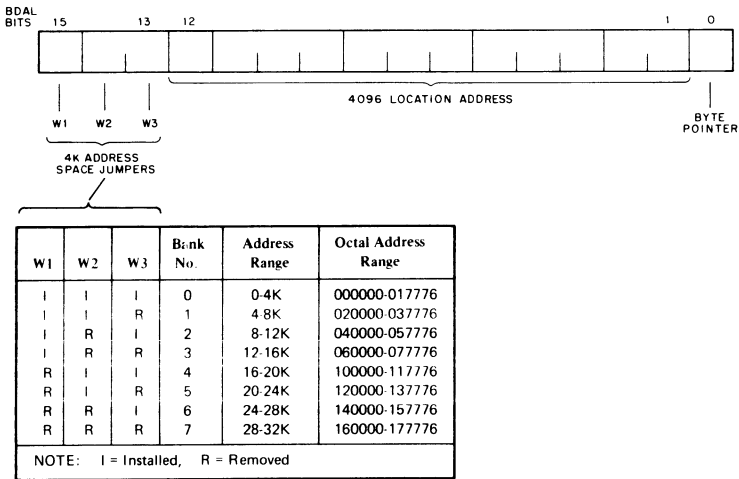


Figure 19-2 MSV11-B Address Format/Jumpers

### 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-CD****MSV11-CD 16K BY 16-BIT MOS READ-WRITE MEMORY**

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 — BENEFITS**

- On-board refresh circuit — eliminates need for refresh signals on the LSI-11 Bus.
- User-configured 4K addresses — allow easy memory layout.
- Refresh jumpers — allow internally or externally controlled refresh.
- Charge pump circuit — only the normal +5 Vdc and +12 Vdc input power required.
- Battery backup jumpers — allow the user to implement battery backup power.

**SPECIFICATIONS****Battery Backup Power**

+5 V 5%	0.8 A
+12 V 3%	0.16 A

**Bus Loads**

AC	2.3
DC	1

<b>System Power</b>	<b>Operating</b>	<b>Standby</b>
+5 V 5%	1.1A	1.1 A
+12 V 3%	0.54 A 12 W	0.16 A 7.7 W

Identification                      M7955-YD

Size                                      Quad

**Access Time**

<b>Bus Cycle Type</b>	<b>Access Time</b>
DATI, DATIO	300 ns typ. (350 ns max. from RSYNC H)
DATO(B)	300 ns typ. (350 ns max. from RDOU 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**

Configuring the MSV11-CD will alter its operation for a specific system application. The following items can be configured:

1. Starting address for the contiguous memory contained on the module
2. Number of banks required on the memory
3. Refresh mode (either on-board or externally generated)
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.

**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 19-3, to the desired starting address as listed in Table 19-1. 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 64 KB 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 19-2 identifies the proper configuration of the jumpers.

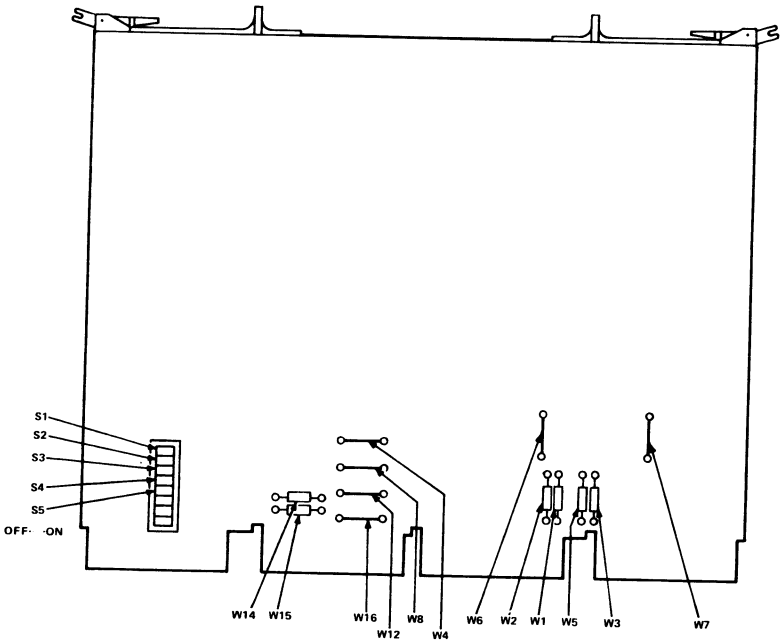


Figure 19-3 MSV11-CD Switch and Jumper Locations

Table 19-1 MSV11-CD Addressing Summary

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.

Table 19-2 Bank Selection

Banks Disabled	W4	W8	W12	W16
None				
3				R
2, 3			R	R
1, 2, 3		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 19-3, provide these functions. W7, when installed, enables internal refresh. W6, when removed, enables the MSV11-CD to reply to external (LSI-11) 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 19-3.

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 19-3 Refresh Mode Selection

Jumper		Refresh Mode Function
W6	W7	
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. These 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 19-3 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.

	<b>1 Module</b>	<b>2 Modules</b>
+Vdc $\pm$ 5%	0.8 A	1.6 A
+12 Vdc $\pm$ 3%	0.18 A	0.32 A

These voltages must remain within  $\pm 3$  percent of the voltages for the LSI-11 Bus at all times and must not change by more than  $\pm 3$  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-D, -E****MSV11-D, -E MEMORY**

All MSV11-D and MSV11-E memory modules can be used in either 16 or 18-bit systems. There are eight versions of this module and they are listed below.

<b>Model</b>	<b>Memory</b>		<b>Parity</b>
	<b>Capacity</b>	<b>Module</b>	<b>Bits</b>
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 — BENEFITS**

- On-board memory refresh — eliminates the need for refresh signals on the LSI-11 Bus.
- User-configured 4K address — allows easy memory layout.
- Charge pump circuit — only the normal +5 Vdc or +12 Vdc input power required.
- Battery backup jumpers — allow the user to implement battery backup power.

**SPECIFICATIONS**

## Identification

MSV11-DX M8044-X  
 MSV11-EX M8045-X (X = A,B,C,D)

Size Double

## Power

	Supply Volt- age	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	0.38 A	0.06 A

	<b>Supply Voltage</b>	<b>Typical Operating Power</b>	<b>Typical Standby Power</b>
	or battery backup		
MSV11-EC,-ED (4K or 16K)	+5V system power	2.0 A	2.0 A
	+5 V battery Backup	1.0 A	1.0 A
	+12 V system power	0.41 A	0.09 A
	or battery backup		
Bus Loading			
AC			2
DC			1
Operational			
MSV11-D			

<b>Bus Cycle Type</b>	<b>Typ</b>	<b>Access Time (ns)</b>			<b>Cycle Time (ns)</b>	
		<b>Max</b>	<b>Notes</b>	<b>Typ</b>	<b>Max</b>	<b>Notes</b>
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

<b>Bus Cycle Type</b>	<b>Typ</b>	<b>Access Time (ns)</b>			<b>Cycle Time (ns)</b>	
		<b>Max</b>	<b>Notes</b>	<b>Typ</b>	<b>Max</b>	<b>Notes</b>
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**

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

**NOTE**

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.

- 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.

### **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 19-4, to the desired starting address as listed in Table 19-4. 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 the 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 19-4, 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:

	<b>MSV11-D (1 Module)</b>	<b>MSV11-E (1 Module)</b>
+5 Vdc $\pm$ 3%	0.7 A	1.0A
+12 Vdc $\pm$ 5%	0.37 A	0.41 A max

These voltages must remain within  $\pm 3\%$  of the bus voltage at all times and not vary more than  $\pm 3\%$  during the transition to or from the battery.

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.

<b>Models</b>	<b>Jumpers (Two Installed)</b>	
	<b>Memory Range Pins</b>	<b>Memory Select Pins</b>
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

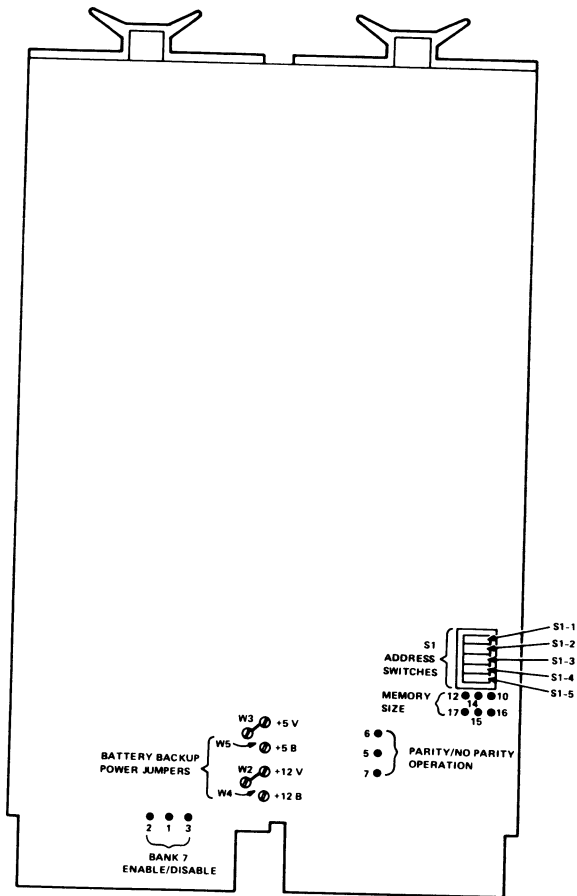


Figure 19-4 MSV11-D, MSV11-E Switch and Jumper Locations



Table 19-4 MSV11-D, MSV11-E Addressing Summary

Starting Address	Switch Settings					Memory Bank(s) Selected					
	S1-1	S1-2	S1-3	S1-4	S1-5	-DA, -EA	-DB, -EB	-DC, -EC	-DD, -ED	-DC, -EC	-DD, -ED
0	C	C	C	C	C	0	0-1	0-3	0-7	0-3	0-7
20000	C	C	C	C	O	1	1-2	1-4	1-10	1-4	1-10
40000	C	C	C	O	C	2	2-3	2-5	2-11	2-5	2-11
60000	C	C	C	O	O	3	3-4	3-6	3-12	3-6	3-12
100000	C	C	O	C	C	4	4-5	4-7	4-13	4-7	4-13
120000	C	C	O	C	O	5	5-6	5-10	5-14	5-10	5-14
140000	C	C	O	O	C	6	6-7	6-11	6-15	6-11	6-15
160000	C	C	O	O	O	7	7-10	7-12	7-16	7-12	7-16
200000	C	O	C	C	C	10	10-11	10-13	10-17	10-13	10-17
220000	C	O	C	C	O	11	11-12	11-14	11-20	11-14	11-20
240000	C	O	C	C	O	12	12-13	12-15	12-21	12-15	12-21
260000	C	O	C	O	O	13	13-14	13-16	13-22	13-16	13-22
300000	C	O	O	C	C	14	14-15	14-17	14-23	14-17	14-23
320000	C	O	O	C	O	15	15-16	15-20	15-24	15-20	15-24
340000	C	O	O	O	C	16	16-17	16-21	16-25	16-21	16-25
360000	C	O	O	O	O	17	17-20	17-22	17-26	17-22	17-26
400000	O	C	C	C	C	20	20-21	20-23	20-27	20-23	20-27
420000	O	C	C	C	O	21	21-22	21-24	21-30	21-24	21-30
440000	O	C	C	O	C	22	22-23	22-25	22-31	22-25	22-31
460000	O	C	C	O	O	23	23-24	23-26	23-32	23-26	23-32
500000	O	C	O	C	C	24	24-25	24-27	24-33	24-27	24-33
520000	O	C	O	C	O	25	25-26	25-30	25-34	25-30	25-34
540000	O	C	O	O	C	26	26-27	26-31	26-35	26-31	26-35
560000	O	C	O	O	O	27	27-30	27-32	27-36	27-32	27-36
600000	O	O	C	C	C	30	30-31	30-33	30-37	30-33	30-37

Starting Address	Switch Settings				Memory Bank(s) Selected								
	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			

## NOTE

- Switch settings  
C = ON  
O = OFF
- Bank 7 cannot be selected as factory-configured; however, the user can enable the lower 2K portion of bank 7 for use.
- X = Do not use.
- 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.

## DESCRIPTION

Logic functions and circuits that comprise MSV11-D and MSV11-E memory modules are shown in Figure 19-5. 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.

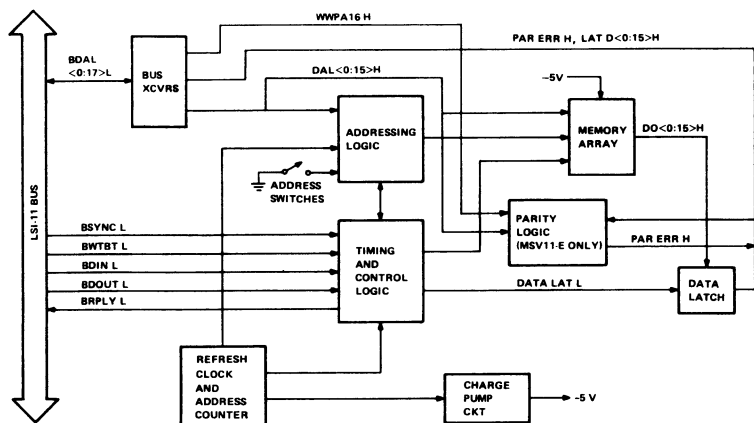


Figure 19-5 MSV11-D and MSV11-E Logic Functions

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	RMA 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

Model	RMA IC Type	Qty RAM ICs in Memory Array	Qty RAM ICs in Parity Logic
MSV11-EB	4K X 1	32	4
MSV-11EC	16K X 1	16	2
MSV-11ED	16K X 1	32	4

The memory array is organized as shown in Figure 19-6. 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 WTBT0 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.

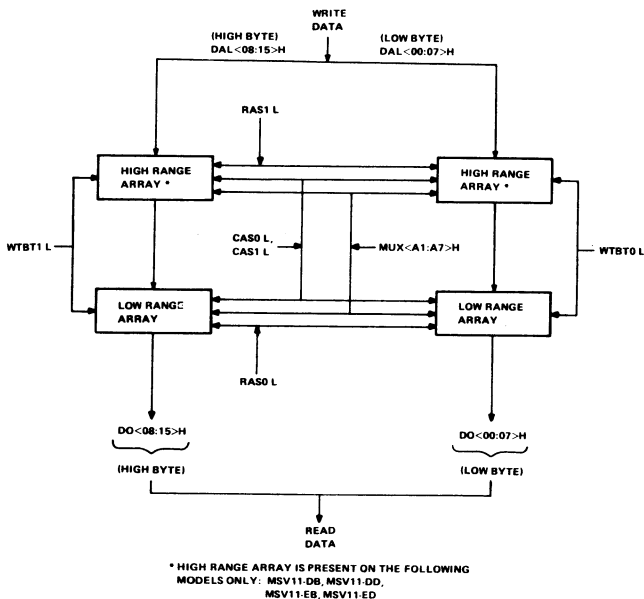


Figure 19-6 Memory Array

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 (CAS1 L and CAS1 L) and row address strobe (RAS1 L and RAS1 L) signals.

### **Addressing Logic**

Addressing logic (Figure 19-7) 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 address 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 RASO 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.

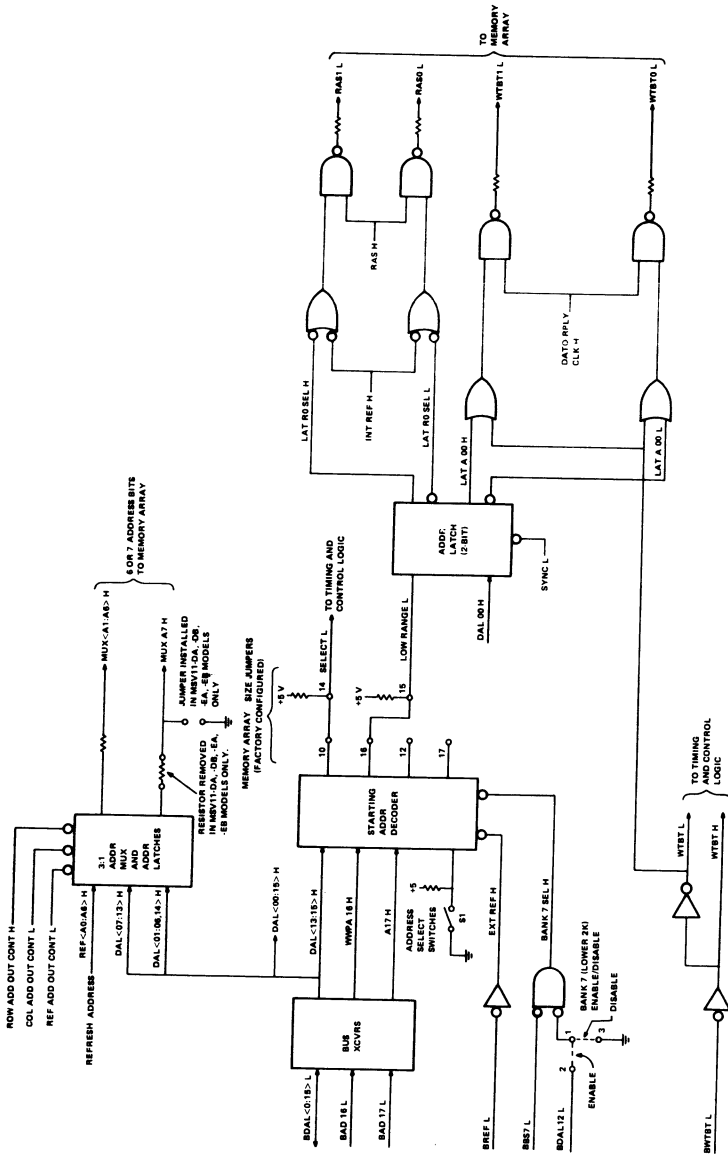


Figure 19-7 Addressing Logic

**Timing and Control Logic**

Circuits comprising timing and control logic are shown in Figure 19-8. Signal sequences for DATI, DATO(B), DATIO(B), and refresh cycles are shown in Figures 19-9 through 19-12, respectively.

The major portions of timing and control functions are contained in the timing and refresh arbitration logic (Figure 19-8). 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.

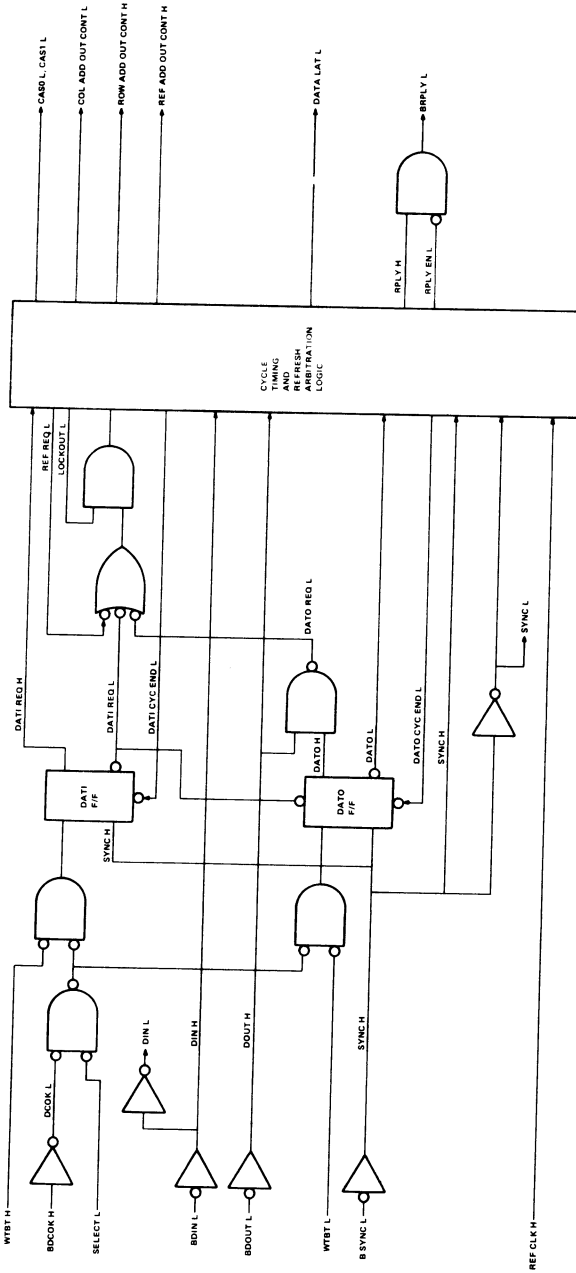


Figure 19-8 Timing and Control Logic



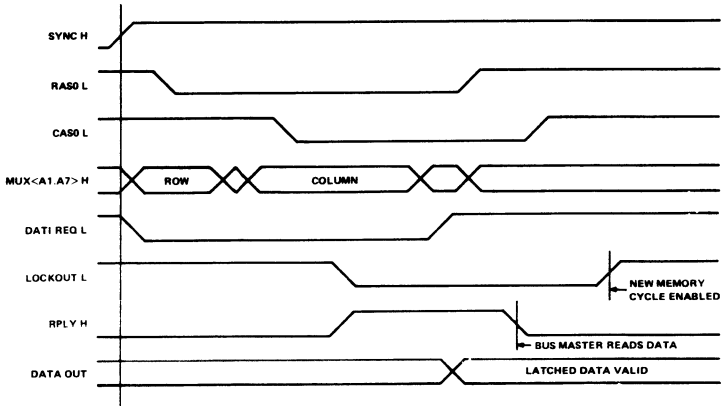


Figure 19-9 DATI Signal Sequence

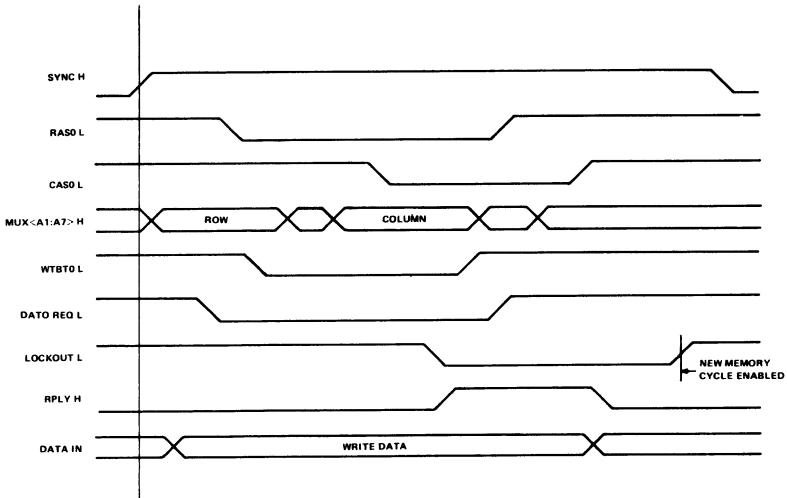


Figure 19-10 DATO(B) Signal Sequence

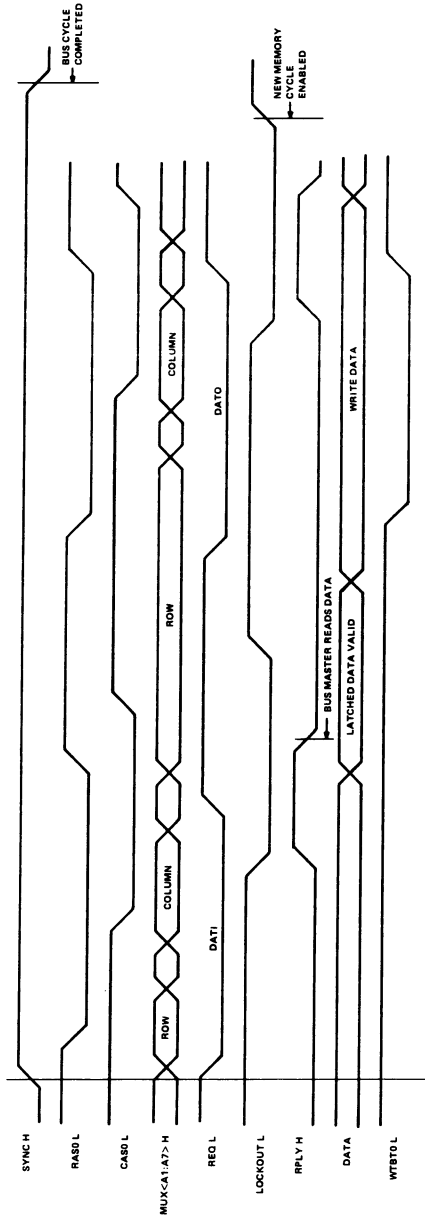


Figure 19-11 DATIO(B) Signal Sequence

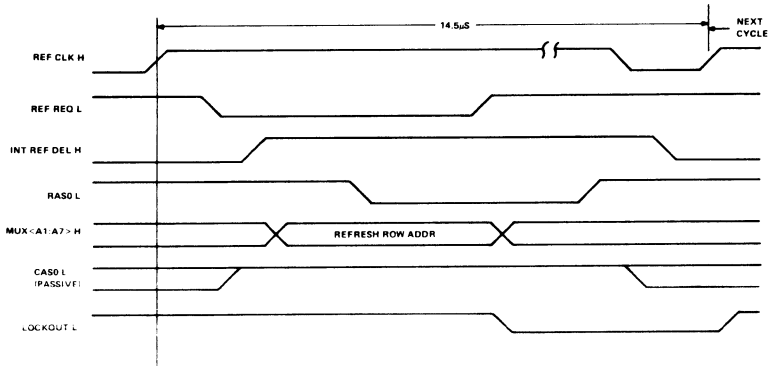


Figure 19-12 Memory Refresh Signal Sequence

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 completed first 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 19-13. A  $14.5\ \mu\text{s}$  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.

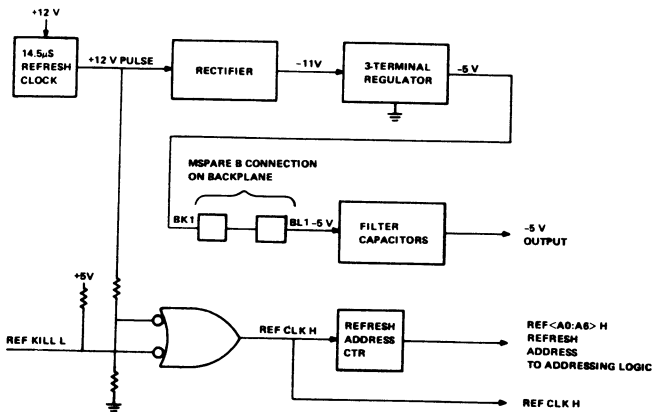


Figure 19-13 Memory Refresh Request Logic

### Charge Pump Circuit

The charge pump circuit produces  $-5\ \text{Vdc}$  for the memory array. Input power is obtained from the  $+12\ \text{V}$  system or battery backup power applied via the refresh clock. The resulting  $12\ \text{V}\ 14.5\ \mu\text{s}$  refresh clock pulse is applied to a rectifier circuit which produces a  $-11\ \text{Vdc}$  (approximately) output. A 3-terminal regulator then produces the required regulated  $-5\ \text{Vdc}$ .

Note that the  $-5\ \text{V}$  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. 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 19-14. 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 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.

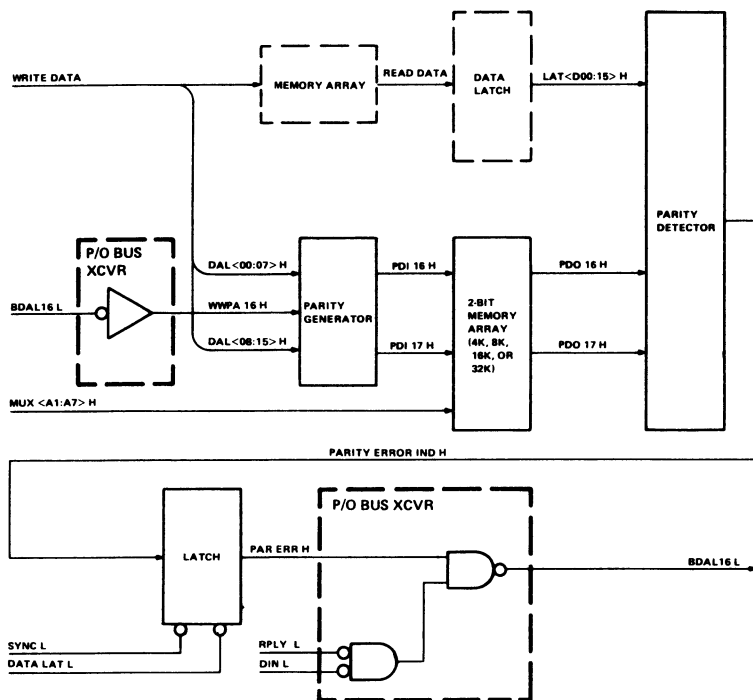


Figure 19-14 Parity Logic

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 16L 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, 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.

**MSV11-L****MSV11-L 128/256 KB MOS RANDOM ACCESS MEMORY**

DIGITAL's MSV11-L family of MOS, random access memory (RAM) modules are the computer industry's first, large-capacity, single-board memories to offer byte parity with 64k X 1 RAM memory technology.

Designed to be used with the LSI-11 Bus, MSV11-L dual-height memory modules provide storage for 18-bit words (16 bits data and 2 parity bits), contain parity control circuitry, and a control and status register (CSR). There are presently two members of the MSV11-L memory module family as shown below.

<b>Model</b>	<b>Module Designation</b>	<b>Storage Capacity</b>
MSV11-LF	M8059-FA	64K by 18 bits
MSV11-LK	M8059-KA	128K by 18 bits

**FEATURES — BENEFITS**

- Dual-height module — totally self-contained, compact modularity.
- Full parity functionality — detects and flags errors on a current cycle (in the event of a failure), eliminates the processing of faulty data, and prevents the execution of faulty code (up to a 2 megabyte system).
- 18- or 22-bit addressing — allows the support of up to 4 megabytes of RAM, per system.

**SPECIFICATIONS**

Size Dual-height 13.2 cm high × 43.9 cm wide × 2.5 cm deep (5.2 in × 8.44 in × .5 in)

Power

	<b>Supply Voltage</b>	<b>Typical Operating Power</b>	<b>Typical Standby Power</b>
MSV11-LF	+5 V ±5% system power	1.45 A	1.40 A

	+5 V battery backup	1.35 A	0.9 A
MSV11-LK	+5 V±5% system power	1.60 A	1.50 A
	+5 V battery Backup	1.40 A	1.0 A

Operational

**Access and Cycle Times\***

Bus Cycles	Notes		Tacc (ns)+		Tcyc (ns)±	
	Tacc+	Tcyc±	Meas Typ	Max	Meas Typ	Max
DATI	2	4	210	230	560	590
DATO(B)	2	5	90	120	605	635
DATIO(B)	3	6	640	670	1140	1170
REFRESH(7)	-	-	-	-	650	685

\* Parity—CSR configurations, refer to notes 1, 8, and 9.

+Tacc = access time

±Tcyc = cycle time

**Notes**

- Assuming memory not busy and no arbitration.
- SYNCH to RPLYH with minimum times (25/50 ns) from SYNCH to (DUNH/DOUTH). The DATO(B) access and cycle times assume a min 50 ns from SYNCH to DOUTH inside memory receivers. For actual LSI-11 Bus measurements, a constant (K-50 ns) where K = 200 ns should be added to DATO(B) times, i.e., Tacc (Typ) = 90 + (200 - 50) = 240 ns.
- SYNCH to RPLYH (DATO(B)) with min time (25 ns) from SYNCH to DINH and minimum (350 ns) from RPLYH (DATI) asserted to DOUT asserted.



4. SYNCH to DL2201 negated.
5. SYNCH to DL2201 negated with minimum time (50 ns) from SYNCH to DOUTH.
6. SYNCH to DL220L (DATO(B)) with min times (25 ns) from SYNCH to DINH and minimum (350 ns) from RPLYH (DATI) asserted to DOUT asserted.
7. Ref REQ L to DL220L negated.
8. REFRESH arbitration adds (100 ns) Typ and (120 ns) Max to access and cycle times.
9. REFRESH conflict adds (650 ns) typ and (685 ns) max to access and cycle times.

### Temperature

#### Storage

–40°C to +66°C (– 40°F to 151°F)

Before operating a module which is at a temperature beyond the operating range, that module must first be brought to an environment within the operating range and then allowed to stabilize for a minimum of five minutes.

#### Operating

+5°C to +60°C (40°F to 140°F)

De-rate the maximum operating temperature by 1°C for each 1000 feet of altitude above 8000 feet.

### Relative Humidity

#### Storage

10% to 90%, non-condensing

#### Operating

10% to 90%, non-condensing

### Airflow, Operating

Adequate airflow must be provided to limit the inlet to outlet temperature rise across the module to 5° when the inlet temperature is +60°C. For operation below +55°C, airflow must be provided to limit the inlet to outlet temperature rise across the module to 10°C maximum.

### Altitude

#### Storage

The module will not be mechanically or electrically damaged at altitudes up to 50,000 feet (90 MM mercury).

**Operating**

Up to 50,000 feet (90 MM mercury). De-rate the maximum operating temperature by 1°C for each 1000 feet of altitude above 8000 feet.

**CONFIGURATION**

The user can configure the following MSV11-L features:

- Module starting address (MSA)
- Control status register (CSR) address
- Battery backup

The MSV11-L module jumpers that are used to configure the board are shown in Figure 19-15, and listed in Tables 19-5 through 19-9. The jumpers listed in Table 19-5 should not need to be changed and are listed only for reference.

**Module Starting Address (MSA)**

The MSA is equal to the decimal K number of words already present in the existing system. Note that all numbers in this section are decimal K values.

Jumpers L, M, N, P, and R select the first address in the 128K block of addresses that contains the module's MSA. Jumpers A, B, C, D, E, and F select the particular 2k increment in that 128K block where the MSA begins. The 128K block selected is called the FAR (for First Address Range). The 2K increment is called the PSA (for Partial Starting Address). The following equation shows how the MSA, FAR, and PSA are related.

$$PSA = MSA - FAR.$$

Refer to Table 19-6 and Table 19-7 for jumper locations for the FAR and PSA respectively.

Taking an MSA of 188K as an example, find the FAR on Table 19-6 by locating the first address in the block with the 188th location. This is 128, the FAR value. To find the PSA, subtract 128 from 188. This yields a PSA of 60. Jumper the FAR of 128 by connecting pin L to pin K (ground). Jumper the PSA of 60 by connecting pin V to pin W to pin X to pin Y to pin U (ground).

**CSR Address Selection**

There are eight addresses reserved for the CSR register. Every MSV11-L module has one CSR. By convention, the memory module with the lowest starting address should be jumpered for the lowest CSR address. The remaining modules should be jumpered in sequence.

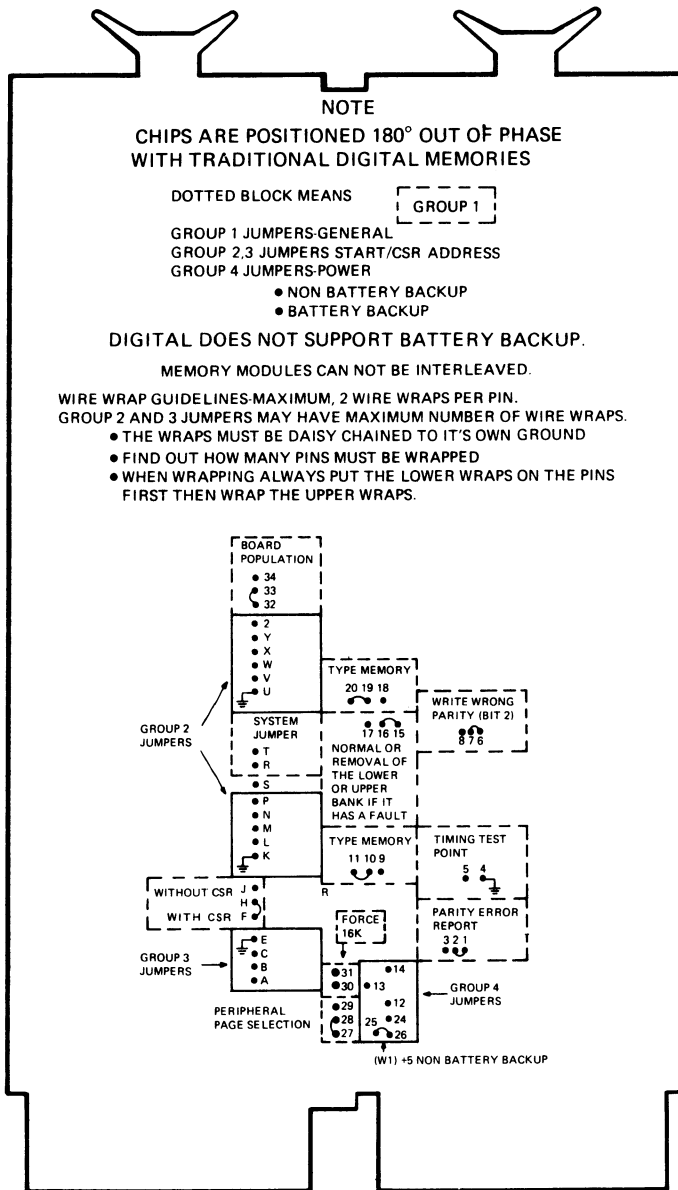


Figure 19-15 MSV11-L Memory Module Jumpers

To select a CSR address for a module, install jumpers according to Table 19-8. Wire-wrap the appropriate pins in daisy-chain fashion to pin E which is grounded.

### Battery Backup

To select either battery backup or no battery backup, jumper the module as shown in Table 19-9.

#### NOTE

DIGITAL does not support battery backup for the MSV11-L.

**Table 19-5 Jumpers and Functions (Group 1)**

<b>Function</b>	<b>Jumper</b>	<b>Normal</b>
<b>Type Memory</b>	<b>Configuration</b>	<b>Condition</b>
Non parity	9 to 10	out
With parity	11 to 10	in
Parity nonCSR	18 to 19	out
Parity with CSR	20 to 19	in
<b>Parity Error Report</b>		
Reported BDAL 16L NonCSR	3 to 2	out
Reported BDAL 16L and BADL 17L with CSR	1 to 2	in
<b>Write Wrong Parity</b>		
Diagnostic bit for tester use:		
Disable	8 to 7	out
Enable	6 to 7	in
<b>CSR Selection</b>		
NonCSR	J to H	out
With CSR	F to H	in
<b>Peripheral Page Selection</b>		
2K peripheral page	29 to 28	out
4K peripheral page	27 to 28	in
<b>Full or One-Half Memory Selection</b>		
Half memory selection	32 to 33	out
Full memory selection	34 to 33	in

Function Type Memory	Jumper Configuration	Normal Condition
<b>Removal of Lower or Upper Bank (with a Fault)</b>		
Lower bank has failed	17 to 16	out
Normal operation or upper bank has failed	15 to 16	in
<b>Extended or Normal Memory Selection</b>		
(Small system) normal operation (128K)	R to T	out
(Large system) extended operation (2 meg. words)	R to T	in

Table 19-6 Starting Address Jumpers (Group 2)

Starting Address Range (FAR)		Jumpers in (X) to Ground (K)				
Decimal (K)	Octal	DAL Pins	21 P	20 N	19 M	18 L
000- 124	00000000- 00760000					
128- 252	01000000- 01760000					X
256- 380	02000000- 02760000				X	
384- 508	03000000- 03760000				X	X
512- 636	04000000- 04760000			X		
640- 764	05000000- 05760000			X		X
768- 892	06000000- 06760000			X	X	
896- 1020	07000000- 07760000			X	X	X
1024- 1148	10000000- 10760000		X			
1152- 1276	11000000- 11760000		X			X

Starting Address Range (FAR)		Jumpers in (X) to Ground (K)				
Decimal (K)	Octal	DAL Pins	21 P	20 N	19 M	18 L
1280-1404	12000000-12760000		X		X	
1408-1532	13000000-13760000		X		X	X
1536-1660	14000000-14760000		X	X		
1664-1788	15000000-15760000		X	X		X
1792-1916	17000000-16760000		X	X	X	
1920-2044	17000000-17760000		X	X	X	

Table 19-7 Starting Address Jumpers (Group 2)

Partial Starting Address (PSA)		Jumpers in (X) to Ground (U)					
Decimal (K)	Octal	DAL Pins	17 Z	16 Y	15 X	14 W	13 V
0	00000000						
4	00020000						X
8	00040000					X	
12	00060000					X	X
16	00100000				X		
20	00120000				X		X
24	00140000				X	X	
28	00160000				X	X	X
32	00200000			X			
36	00220000			X			X
40	00240000			X		X	
44	00260000			X		X	X
48	00300000			X	X		
52	00320000			X	X		X
56	00340000			X	X	X	
60	00360000			X	X	X	X

Partial Starting Address (PSA)		Jumpers in (X) to Ground (U)					
Decimal (K)	Octal	DAL Pins	17 Z	16 Y	15 X	14 W	13 V
64	00400000		X				
68	00420000		X				X
72	00440000		X			X	
76	00460000		X			X	X
80	00500000		X		X		
84	00520000		X		X		X
88	00540000		X		X	X	
92	00560000		X		X	X	X
96	00600000		X	X			
100	00620000		X	X			X
104	00640000		X	X		X	
108	00660000		X	X		X	X
112	00700000		X	X	X		
116	00720000		X	X	X		X
120	00740000		X	X	X	X	
124	00760000		X	X	X	X	X

Table 19-8 CSR Address Jumpers (Group 3)

22 Bit CSR Address	18 Bit CSR Address	C	B	A
17772100	772100			
17772102	772102			X
17772104	772104		X	
17772106	772106		X	X
17772110	772110	X		
17772112	772112	X		X
17772114	772114	X	X	
17772116	772116	X	X	X

Table 19-9 Power Jumpers (Group 4)

Voltage Connection	Jumper Configuration
+5V V Nonbattery backup	26 to 25 (W1)

Voltage Connection	Jumper Configuration
+5 V Battery backup	25 to 25 (W2) 14 to 13 (W3)* or 12 to 13 (W4)*

\* Availability for the +5 V battery backup

**Table 19-10 MSV11-L Jumper Check List**

Jumper Name or Pin to Pin	Jumper In	Jumper Out	Wire Wrap	Solder	Check
General Jumpers (Group 1)					
9 to 10	X		X		
11 to 10		X	X		
18 to 19		X	X		
20 to 19	X		X		
3 to 2		X	X		
1 to 2	X		X		
8 to 7		X	X		
6 to 7	X		X		
J to H		X	X		
F to H	X		X		
20 to 28		X	X		
27 to 28	X		X		
32 to 33			X		
34 to 33			X		
17 to 16		X	X		
14 to 16	X		X		
R to T		X	X		
Starting Address Jumpers (Group 2)					
P		X	X		
N		X	X		
M		X	X		
L		X	X		
T		X	X		
K*					
Z		X	X		
Y		X	X		
X		X	X		
W		X	X		



Jumper Name or Pin to Pin	Jumper In	Jumper Out	Wire Wrap	Solder	Check
V		X	X		
U*					

## Control and Status Register Jumpers (Group 3)

C		X	X		
B		X	X		
A		X	X		
E*					

## Power Jumpers (Group 4)

W1(25 to 26)	X				X
W2(24 to 25)		X			
W3(13 to 14)		X			
W4(12 to 13)		X			

\* Ground Pin

**DESCRIPTION**

The MSV11-L memory module consists of a single, dual-height module that contains the LSI-11 Bus interface, timing and control logic, refresh circuitry, and a MOS storage array. The module also contains circuitry to generate and check the parity, and a control and status register. Figure 19-16 is a block diagram of the MSV11-L.

**Address Logic**

Address logic is divided into these sections:

- Module selection decode
- MOS memory address logic

**Module Selection Decode**—Module Selection is accomplished by 2 PROMs, that are jumpered for the starting address of the module on 4K boundaries. The PROMs compare bus data address lines (BDAL) 21-13 against the starting address jumpers. If an address on these lines is within the range of addresses that belong to the module, the PROM will enable the memory module to be selected and specify what row the address is in.\*

**MOS Memory Address Logic**—The MSV11-L memory module performs three functions:

- Memory read/write
- CSR read/write
- Refresh writes

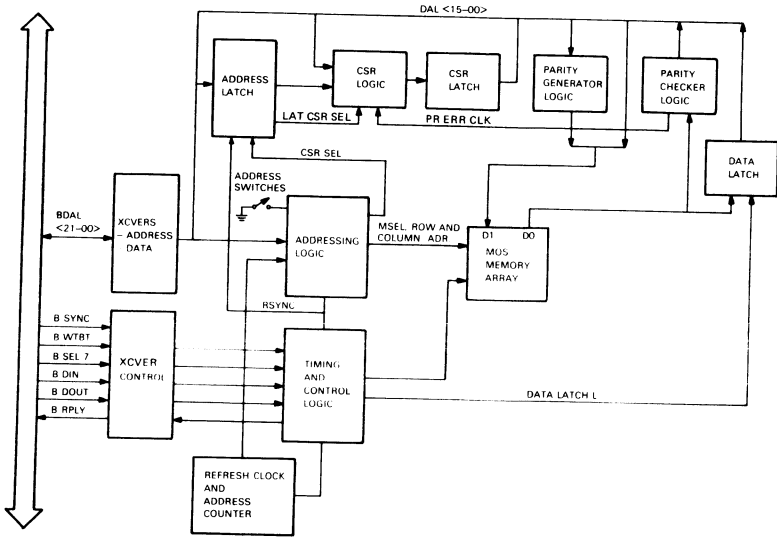


Figure 19-16 MSV11-L Block Diagram

Figure 19-17 shows how the address logic handles the three functions. Memory and CSR references have their address latched into the row and column address latches. Refresh cycles use the refresh latch to obtain an address.

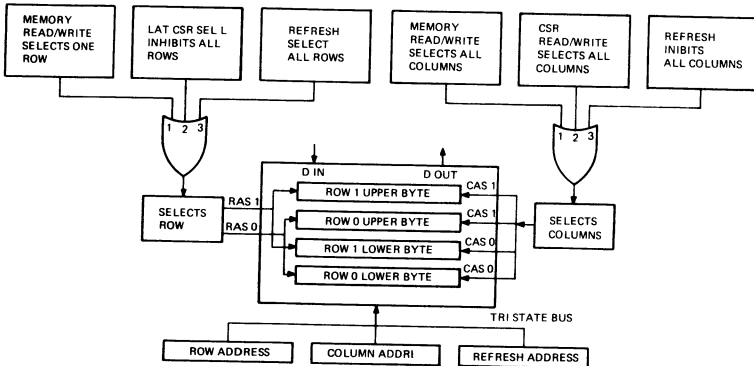


Figure 19-17 MOS Memory Address Logic

**NOTE**

For CSR references, RAS is never selected during CSR cycles, so row and column selection does not matter.

Control and timing enable the memory module to load the row address via row address strobe (RAS) and column address via column address strobe (CAS) into the MOS memory. The memory then reads or writes, depending on the command issued by the master.

Memory read/write requires selection of a row (RAS0 or RAS1) and of all the columns.

CSR read/write inhibits all rows. Therefore no data can be read or written to MOS memory, but will be read or written to the CSR register. All columns will be selected, with no effect.

Refresh selects all rows and inhibits all columns.

**Row Selection**

The MSV11-LK memory module is a fully-populated module (128K words). The signal from the PROM, RAS SEL (L), is active for the first 64K words of address space. The signal from the PROM for the second 64K words of address, RAS SEL H, is negated.

The signals that will be generated are as follows. RAS SEL (L) will cause SEL RAS 0 to be active. Row address strobe 0 (RAS0) gates the row address into row 0 memory chips (Figure 19-18).

RAS SEL (H) will cause SEL RAS 1 to be active. Row address strobe 1 (RAS1) gates the row address into row 1 memory chips (Figure 19-18).

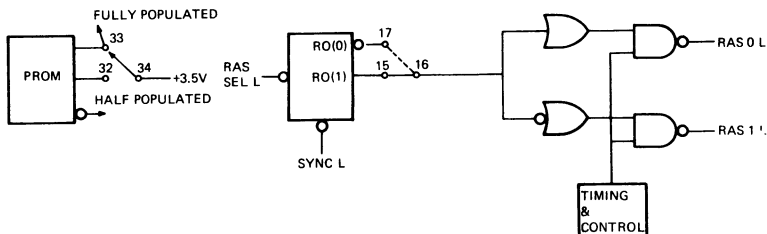


Figure 19-18 Row Selection

If a fully-populated memory module has bad memory in one of the two 64K words sections, the module can be forced to act like a half-populated module by setting jumpers 15 to 16 to 17. In either case, the

address will be from 0K to 64K words. This means that RAS SEL L will be active whenever the good 64K chips are addressed.

For example, if the first 64K words of memory have bad memory chips, jumper pin 16 to pin 17. This enables the logic to generate RAS1. Remember that RAS SEL (L) will be active (Figure 19-18).

If the second 64K words of memory have bad memory chips, jumper pin 15 to pin 16. This enables the logic to generate RAS0. Remember that RAS SEL (L) is active (Figure 19-18).

### Memory Array

The MOS chips used in the MSV11-LF/LK memories are dynamic random access memory circuits organized as a 65,536 by 1 bit chip. The chip requires 128 refresh cycles within 2 ms (Figure 19-19).

The memory module has two configurations.

1. Fully-populated
  - 2 rows of 18 MOS chips
    - 9 chips handle low byte and parity
    - 9 chips handle high byte and parity
2. Half-populated
  - 1 row of 18 MOS chips
    - 9 chips handle low byte and parity
    - 9 chips handle high byte and parity

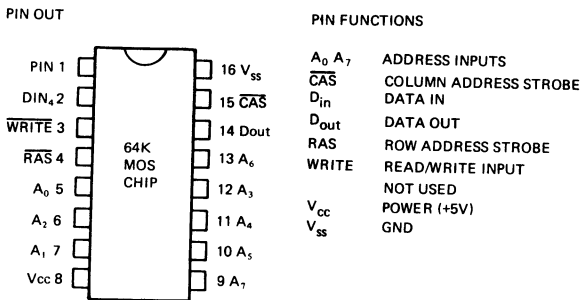


Figure 19-19 64K MOS Ram Chips

### Parity Logic

The parity logic has two functions.

1. Parity generation when the bus master is doing a DATO(B)
2. Parity checking when the bus master is doing a DATI

**Parity Generation**—When memory receives the data from the bus master, DAL 15-00 goes to the parity generators (Figure 19-20). DAL 07-00 and PDI16H generate odd parity when writing into memory. DAL 15-08 and PDI17H generate even parity when writing into memory.

CSR 02 (H) is used for diagnostic purposes and enables the diagnostic to force wrong parity when the data word is stored in memory.

To use DAL16 with a tester on data transfers to force wrong parity, jumper pin 6 to pin 7. DAL16 should never be active at data time when the system is in normal use.

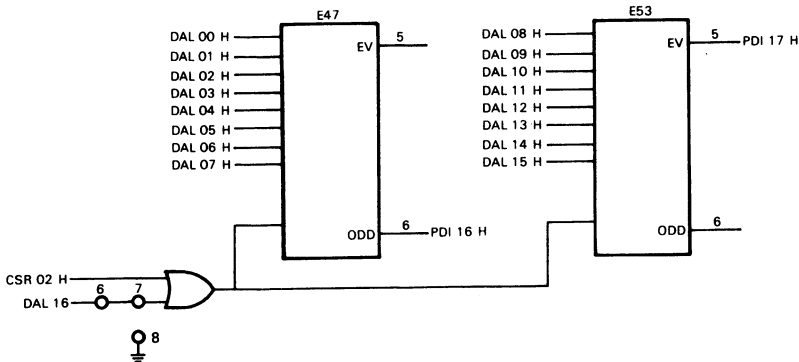


Figure 19-20 Parity Generators

**Parity Checking**—The MSV11-L memory modules detect, report, and save the address of parity errors. Parity detection logic (Figure 19-21) on reads expects byte 0 to have an even number of 1 bits and byte 1 to have an odd number of 1 bits. If this does not happen, a parity error is detected. The parity error signal is called LAT PAR ERR.

Parity reporting (Figure 19-22) shows parity errors are reported by BDAL 16 and 17, both being active. This means that a jumper connects pin 1 to pin 2 and CSR bit 15 is set.

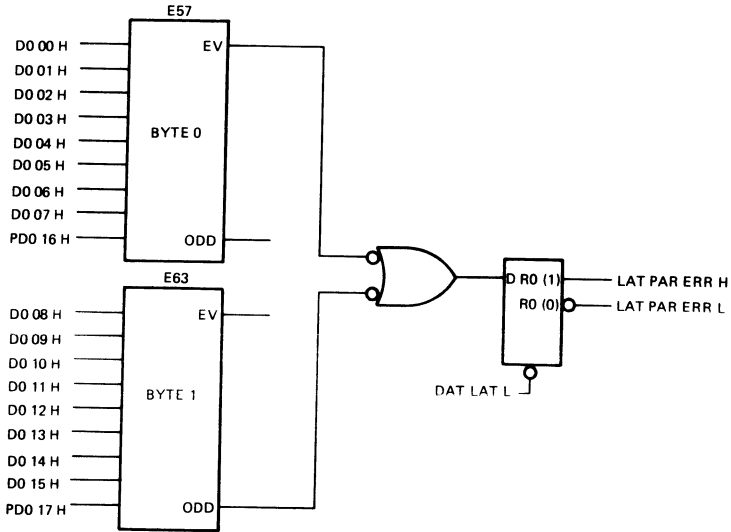


Figure 19-21 Parity Detection

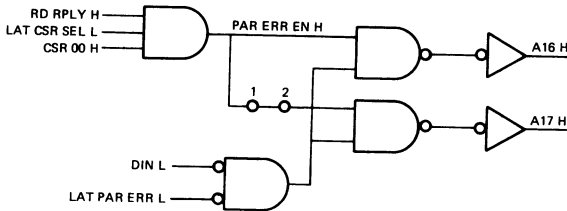


Figure 19-22 Parity Reporting

Everytime a bus master references memory, address bits BDAL 21-11 are latched. If a parity error occurs, CSR CLK and PAR ERR CLK latch the address in LAT D05-D011. LAT A18-A21 will be stored in LAT D05-D08 (Figure 19-23).

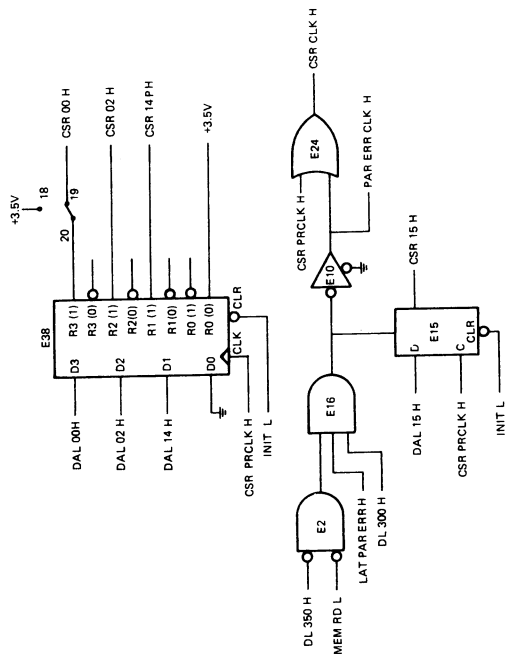
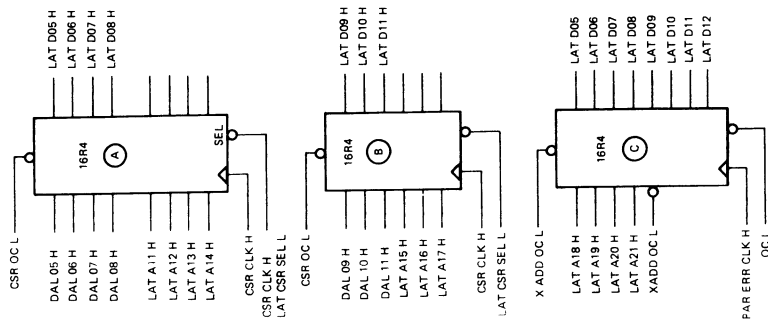


Figure 19-23 Address of Parity Error

### Timing and Control Logic

This logic responds to read/write cycles and refresh cycles. The read/write request from a bus master references the memory (storage) or the CSR register.

Refresh requests are generated every  $14.5 \mu\text{s}$  from within the memory module. The read/write and/or refresh request will generate a memory request (Figure 19-24). If memory is not locked out, arbitration takes place between the requests that are present. The arbitration circuit decides who will use the memory. The loser waits for a turn.

Figure 19-25 is the memory flow diagram. Figures 19-26, 19-27, 19-28, 19-29 show memory timing.

### Refresh Cycle

Refresh requests are initiated every  $14.5 \mu\text{s}$  by generating REF REQ (L). REF REQ (L) does two things:

1. It arbitrates for the use of memory timing when memory lockout is negated.
2. It increments the refresh counter.

If the REF flip-flop gets set prior to DLO, arbitration was won by the refresh request.

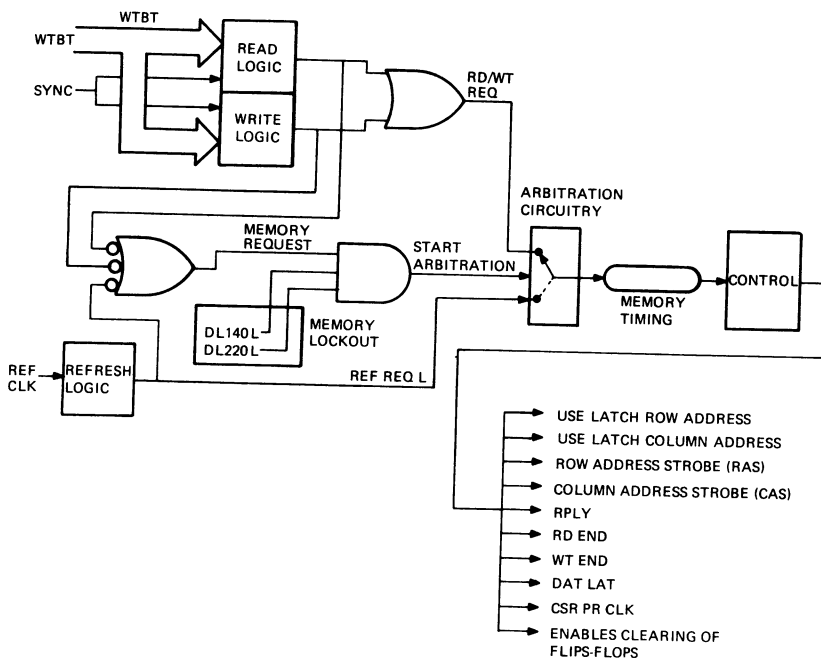


Figure 19-24 Timing and Control





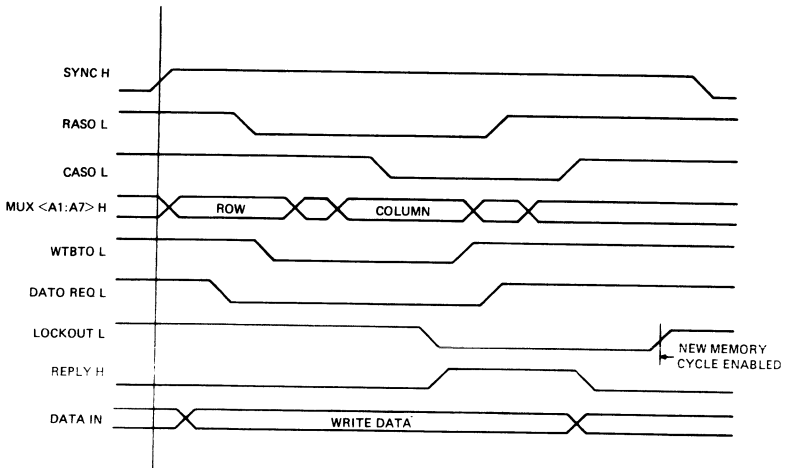


Figure 19-26 DATO(B) Signal Sequence

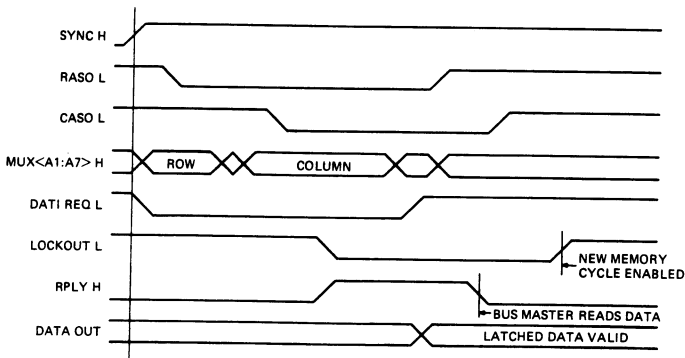


Figure 19-27 DATI Signal Sequence

The refresh address is placed on TRI-STATE BUS and Row Address Strobe (RASO and RAS1) gates the address into the MOS chips. Column Address Strobe (CAS) is inhibited. The MOS chips are then refreshed internally.

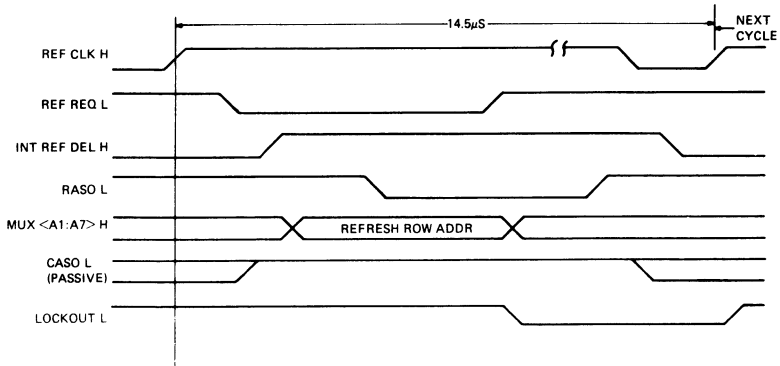


Figure 19-28 DATIO(B) Signal Sequence

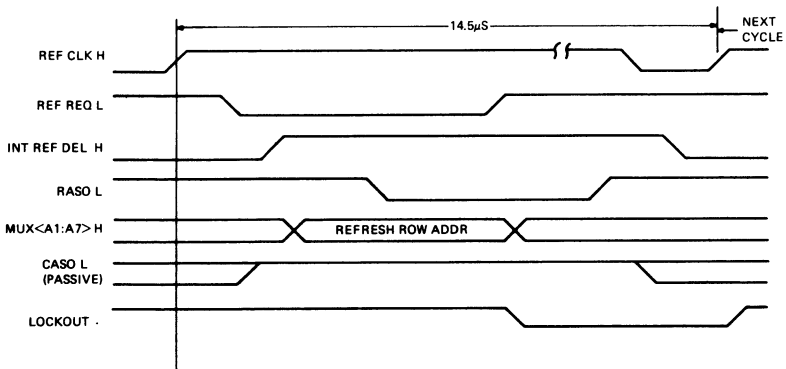


Figure 19-29 Memory Refresh Signal Sequence

### Control and Status Register

The control and status register in the MSV11-L memory module has the following features:

- It holds parity error address
- It forces wrong parity
- It enables parity error reporting
- It has a parity error flag
- It can enable extended error address reporting for large systems

The CSR has a reserved address in the top 4 K of memory. To reserve the CSR address, the memory module is jumpered. The CSR allows program control of certain parity functions and contains diagnostic information if a parity error has occurred. The CSR is assigned an address and can be accessed by a bus master via the LSI-11 Bus. Some CSR bits are cleared by the assertion of BUS INIT (L). This signal is asserted for a short period of time by the processor after the system power has come up, or in response to a reset instruction. For more detailed information on CSR bit assignments, please consult the MSV11-L User Guide--EK-MSVOL-UG-001.

The CSR bits have the following functions:

Bits 1, 3, 4      **Not used.** These bits are always read as logical 1's. Writing to these bits has no effect on the CSR.

Bit 0      **Parity Error Enable.** If set, and a parity error occurs on a DATI or DATIO(B) cycle to memory, then BDAL 16 (L) and BDAL 17 (L) will be asserted on the bus at the same time as the data. The asserted lines are selected by jumpers. This bit is read/write, and is reset to zero on power-up or BUS INIT.

Bit 2      **Write Wrong Parity.** If set, and a DATO spare or DATOB cycle to memory occurs, wrong parity data will be written into the parity MOS RAMs. This bit may be used to check the parity error logic as well as failed address information in the CSR. Use this bit to test the parity logic as follows:

1. With bit 2 set, write entire memory with any pattern.
2. Read first location in memory. If bit 0 of the CSR is set, then a parity error should be detected on the LSI Bus and the failed address (0) should be stored in the CSR.
3. Read the CSR and obtain the failed address. If CSR bit 14 equals 1, then bits A11-A17 are loaded into CSR bits 5-11. If CSR bit 15 equals 1, then bits A18-21 have been loaded into CSR bits 5-8. Bit 2 is a read/write bit reset to zero on power-up or BUS INIT.

Bits 05-11      **Error Address Bits.** If a parity error occurs on a DATI or DATIO(B) cycle, then A11-A17 are stored

in CSR bits 5-11 and bits A18-A21 are latched. The 18-bit address machines require only one read of the CSR to obtain the failed address. The CSR bit set to 0 allows the logic to pass A11-A17 to the LSI-11 Bus. A 22-bit machine requires two reads. The first read (with CSR bit-14 set to 0) sends contents to CSR bits 5-11. Then the software must set CSR bit-14 equal to 1 to enable A18-A21 to be read from CSR bits 5-8. The parity error address locates a parity error to a 1K segment of memory.

These are read/write bits and are not reset to zero via power-up or BUS INIT. If a second parity error is found, the new failed address will be stored in the CSR.

- Bit 14      **Extended CSR Read Enable.** This bit is always read as a 0 on 18-bit address machines. On 22-bit address machines, bit 14 is set to 1 by software on the second CSR read of a failed memory address. This allows retrieval of failed address bits A18-A21, stored in CSR bits 5-8. This bit is reset to 0 on power-up or BUS INIT.
- Bit 15      **Parity Error Detect.** If a parity error occurs on a DAT1 or DAT10(B) cycle, this bit will be set to 1. This is a read/write bit and is reset to 0 by power-up or BUS INIT. Bit 15 will remain set unless rewritten or initialized.

### CSR Logic Overview

The memory module receives a CSR address that is compared with the CSR address jumpers. The CSR register is selected if there is a match. Bus signal BWTBT then determines whether the memory will do a read or a write operation. When the signal BSYNC arrives, the decision is final.

The chips that comprise the CSR register are E38 and PAL chips E50, E58, E71.

### CSR Write

There are two parts to a CSR Write operation; address decode and a data cycle. During address decode, the CSR SEL signal enables the write request flip-flop to get set when BSYNC is received. During the data cycle, the memory module receives the data on the BDAL lines.

The control signal BDOUT enables the write request to go to arbitration. When allowed, the write request will start memory timing. During the timing phase, the following conditions exist.

- Latch CSR select
- Refresh negated
- Read request negated

The result is that the CSR clock logic gates the received data into the CSR register.

### CSR Read

A CSR Read operation also consists of an address decode portion and a data cycle portion. During address decode, the signal CSR SEL enables the Rd Req flip-flop. Two events occur when BSYNC is received:

1. The Rd Req flip-flop is set.
2. BDAL 21-11 and BDAL00 get latched, although this has no effect. The read request then goes to arbitration and starts memory timing, when allowed.

During the data cycle, memory sends back the contents of the CSR register. The data received by the CPU may be parity information or diagnostic information that is used to check out the integrity of the CSR.

A diagnostic read requires a known pattern to be written into the CSR register. Programmable Logic Arrays (PALs) have DAL bits (DATA INPUT) selected as a result of the signal LAT CSR SEL (L); when CSR CLK occurs, the data will be transferred into the CSR.

On reads, the signals CSROC(L) and OC (L) enable the CSR data (LAT DO bits) to be output to the memory transmitters.

A parity error during a read operation requires that the CPU read the address of the parity error. Both small and large systems read the contents of the CSR register.

Small systems keep CSR bit 14 equal to 0 and write latched address bits A17-A11 into the CSR register (PAL). The contents of the CSR are gated out of the PAL, when the signal CSR OC (L) and OC (L) is active. The signals A17-A11 become LATDO11-LATDO05.

Large systems require 2 reads to receive A21-A11. For the first read CSR, bit 14 equals 0. This enables the memory to pass A17-A11 from CSR bits 11-05. For the second read, software must do a CSR write to set CSR bit 14.

Then the system does the second read. CSR bit 14 equals 1, enables memory to pass bits A21-A18 from CSR bits 08-05.

**MCV11-D****MCV11-D 8K BY 32 KB CMOS READ/WRITE MEMORY WITH BATTERY BACKUP**

The MCV11-D is a 32 KB static CMOS read/write memory with battery backup that can be installed on an LSI-11 backplane. The MCV11-D memory is available in two versions: the MCV11-DA and the MCV11-DC. The MCV11-DA and the MCV11-DC options have a storage capacity of 8K and 32 KB, respectively. Data can be stored in either a word or byte format.

The MCV11-D consists of a single, dual-height module that contains the LSI-11 Bus interface, timing, and control logic, CMOS storage array, and two 1.2 Volt rechargeable batteries. The module's starting address is jumper selectable on any 4K word boundary within the system's address space. The upper 4K words of address space are reserved for an I/O page. However, the module can be jumpered to permit use of the lower 2K words of the I/O page as memory locations.

**FEATURES — BENEFITS**

- Fast access time — 250 ns maximum.
- Battery Backup — protects data when power is removed.
- Lower power — requires a smaller power supply and less cooling.
- Static operation — requires no refresh logic.
- User-configured 4K word boundary addresses — allow flexibility of memory layout.

**SPECIFICATIONS****Power****MCV11-DC (32 KB)**

		<b>Active Mode</b>	<b>Standby Mode</b>	<b>Data Retention Mode</b>	<b>Notes</b>
	+5 V Typ	1.23 A	1.22 A	0	(1)
	+5 V Max	2.16 A	2.15 A	0	(1)
Current	+5 V BBU Typ	1 mA	1 mA	9 mA	(2)
	+5 V BBU Max	2 mA	2 mA	14 mA	(2)
Power	+5 V Typ	6.2 W	6.1 W	.045 W	
	+5 V Max	11.34 W	11.29 W	.073 W	

**MCV11-DA (8 KB)**

	+5 V Typ	1.20 A	1.19 A	0	(1)
	+5 V Max	2.09 A	2.08 A	0	(1)
Current	+5 V	1 mA	1 mA	9 mA	(2)
	BBU Typ				
	+5 V	2 mA	2 mA	14 mA	(2)
	BBU Max				
Power	+5 V Typ	6.0 W	5.95 W	.045 W	
	+5 V Max	10.97 W	10.92 W	.073 W	

**Notes**

- +5 V current is recorded with no +5 V BBU supply connected.
- +5 BBU current assumes +5 V equals 4.75 V and +5 BBU equals 5.25 V. In the active and standby mode, a majority of current comes from the +5 V supply, so that it appears as though very little current is required by the +5 BBU supply. In the data retention mode, the +5 V supply is assumed to be at 0 V. The current supplied by +5 BBU is used to trickle charge the batteries. If the batteries were disconnected, +5 BBU would be typically 20  $\mu$ A.

**Data Retention**

Option	Size	Worst Case	
		Typical Data Retention	Data Retention
MCV11-DC	32 KB	1180 hr.	100 hrs.
MCV11-DA	8 KB	2647 hr.	333 hrs.

**Access and Cycle Times**

Bus Cycles	Notes		T <sub>acc</sub> (ns)+		T <sub>cyc</sub> (ns) $\pm$	
	T <sub>acc</sub> +	T <sub>cyc</sub> $\pm$	Meas Typ	Max	Meas Typ	Max
DATI	1	4	225	250	520	570
DATO(B)	2	4	50	55	500	550
DATIO(B)	3	5	590	620	1010	1070



**Notes**

1. RSYNC H to T RPLY H with minimum timing (25 ns) from RSYNC H to R DIN H and typical or maximum module propagation delays.
2. R SYNC H to T RPLY H with minimum timing (50 ns) from R SYNC H to R DOUT H typical or maximum module propagation delays.
3. R SYNC H to T RPLY H (DATO portion of the bus cycle) with minimum timing (25 ns) from R SYNC H to R DIN H and minimum timing (350 ns) from T RPLY H (DATI portion of cycle) to R DOUT H.
4. R SYNC H to TIM 130 H negated.
5. R SYNC H to TIM 130 H negated (DATO).

**Temperature****Storage**

−30°C to +60°C (−30°F to +108°F)

Before operating a module which has been at a temperature beyond the operating range, that module must first be brought to an environment within the operating range and then allowed to stabilize for a minimum of five minutes.

Battery exposure to high temperature (greater than 50°C) will speed up self discharge. Operating the module outside the recommended temperature range may cause battery failure.

**Operating**

5°C to 60°C (40°F to 140°F)

De-rate the maximum operating temperature by 1°C for each 1000 feet of altitude above 8000 feet.

**Relative Humidity****Storage**

10% to 90%, noncondensing

**Operating**

10% to 90%, non-condensing

**Airflow, Operating**

Adequate airflow must be provided to limit the inlet to outlet temperature rise across the module to 5°C (41°F) when the inlet temperature is 55°C (99°F). For operation below 50°C (122°F), airflow must be provided to limit the inlet to outlet temperature rise across the module to 10°C (18°F) maximum.

**Altitude****Storage**

The module will not be mechanically or electrically damaged at altitudes up to 50,000 feet (90 MM mercury).

**Operating**

Up to 50,000 feet (90 MM mercury). De-rate the maximum operating temperature by 1°C for each 1000 feet of altitude above 8000 feet.

**CONFIGURATION**

The user can configure the following MCV11-D features:

- Module starting address
- 16, 18, or 22-bit addressing
- Memory I/O page size
- Battery backup

**Module Starting Address (MSA)**

The Module starting address is equal to the decimal K number of words already present in the existing system. Note that all numbers in this section are decimal K values.

Jumpers L, M, N, P, and R select the first address in the 128K word block of addresses that contains the module's MSA. Jumpers A, B, C, D, E, and F select the particular 4K word increment in that 128K block where the MSA begins. The 128K block selected is called the FAR (for First Address Range). The 2K increment is called the PSA (for Partial Starting Address). The following equation shows how the MSA, FAR and PSA are related:

$$PSA = MSA - FAR.$$

Refer to Table 19-11 and Table 19-12 for jumper locations for the FAR and PSA, respectively.

**Table 19-11 FAR Jumper Locations**

First Starting Address Range (FAR)		Jumpers In (X) to Ground (R)				
		Decimal (K)	Octal	L	M	N
000—124	00000000—00760000					
128—252	01000000—01760000					X
256—380	02000000—02760000					X
384—508	03000000—03760000					X
512—636	04000000—04760000			X		
640—764	05000000—05760000			X		X

First Starting Address Range (FAR) Decimal (K)	Octal	Jumpers In (X) to Ground (R)			
		L	M	N	P
768—892	06000000—06760000		X	X	
896—1020	07000000—07760000		X	X	X
1024—1148	10000000—10760000	X			
1152—1276	11000000—11760000	X			X
1280—1404	12000000—12760000	X		X	
1408—1532	13000000—13760000	X		X	X
1536—1660	14000000—14760000	X	X		
1664—1788	15000000—15760000	X	X		X
1792—1916	16000000—16760000	X	X	X	
1920—2044	17000000—17760000	X	X	X	X

**NOTE**

The MCV11-D is shipped as a small system (i.e. 32K word system). For applications where this memory is located outside this address space, the memory must be configured as a large system. (wire-wrap J to R).

**Table 19-12 PSA Jumper Locations**

Partial/Starting Address (PSA)		Jumpers in (X) To Ground (F)				
Decimal (K)	Octal	A	B	C	D	E
0	00000000					
4	00020000					X
8	00040000				X	
12	00060000				X	X
16	00100000			X		
20	12			X		X
24	14			X	X	
28	16			X	X	X
32	20		X			
36	22		X			X
40	24		X		X	
44	26		X		X	X
48	30		X	X		
52	32		X	X		X

Partial/Starting Address (PSA)		Jumpers in (X) To Ground (F)				
Decimal (K)	Octal	A	B	C	D	E
56	34		X	X	X	
60	36		X	X	X	X
64	40	X				
68	42	X				X
72	44	X			X	
76	46	X			X	X
80	50	X		X		
84	52	X		X		X
88	54	X		X	X	
92	56	X		X	X	X
96	60	X	X			
100	62	X	X			X
104	64	X	X		X	
108	66	X	X		X	X
112	70	X	X	X		
116	72	X	X	X		X
120	74	X	X	X	X	
124	00760000	X	X	X	X	X

Taking an MSA of 336K word as an example, find the FAR on Table 19-11 by locating the first address in the block with the 336th location. This is 256, the FAR value. To find the PSA, subtract 256 from 336. This yields a PSA of 80. Jumper the FAR of 256 by connecting pin N to pin R (ground). Jumper the PSA of 80 by connecting pin A to pin C to pin F (ground).

### Selecting 16, 18, or 22-bit Addressing

The MCV11-D will support either 16, 18, or 22-bit addressing. To select 16 or 18-bit addressing, remove the jumper at pin J. To select 22-bit addressing, connect pin J to R (ground).

### Modifying the I/O Page

The top 4K words of address space is usually reserved to address I/O devices. The user may modify the I/O page by jumpering pin U to pin V. This will add the bottom 2K words of the I/O page to memory.

### Enabling Battery Backup

The MCV11-D module comes with two batteries already installed. To enable the battery backup function, remove the clip across pins Y and Z and connect it to pins W and X.

The two batteries included with the module are rechargeable nickel cadmium cylindrical cells. Each cell is size AAA, 1.2 V nominal, with a 180 mA hr capacity at 25°C. The battery operating life is projected at over five years at normal operating temperatures. If the operating temperatures are continuously high, then battery life will be shortened substantially. Battery life is sustained for approximately one year in a 60°C environment. It is recommended that the module be kept in operation until the battery fails to hold a charge sufficient for the system's application. Only in the most demanding situations, where a data retention loss proves very costly, should a battery replacement schedule be advised. In these situations, the use of an external remote monitored power source on the BBU pin is recommended.

The charge rate for the batteries is approximately 12 mA as long as +5 V is present. It takes 24 hours to totally charge a fully discharged battery. For every hour of charging, approximately 1/24th of the total charge will be replaced.

## DESCRIPTION

The MCV11-D is an LSI-11 Bus device, and supports the DATI, DATO(B), and DATIO(B) cycles described in Chapter 9, the LSI-11 Bus chapter. Figure 19-30 is a block diagram of MCV11-D functions.

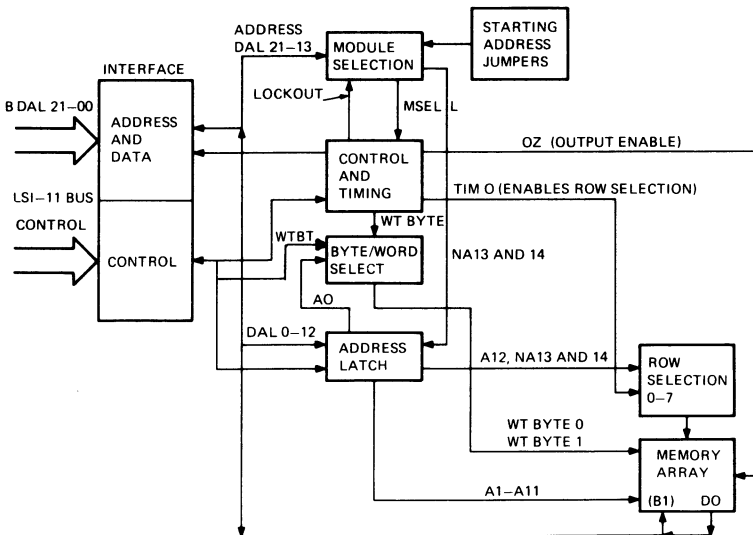


Figure 19-30 MCV11-D Block Diagram

The major functions on the MCV11-D are as follows:

- Interface
- Module Select
- Address Latch
- Control and Timing
- Byte/Word Select
- Memory Array

**Interface**—The interface connects the MCV11-D to the LSI-11 Bus via transceivers. The interface consists of a data/address section and a control section. The address section receives address signals on bus data address lines (BDAL) 21-00 and sends data signals on BDAL 15-00.

The interface monitors the address lines and sends DAL bits 21-13 to the module selection block. DAL bits 12-00 will be sent to the address latch if the memory module has been selected. The data is transferred from BDAL 15-00 to the memory array during memory writes. Data is transferred from the memory array to BDAL 15-00 on memory reads.

The control section of the interface passes control signals from the bus to the control and timing circuits. The WTBT (write byte) signal is sent to the address latch and the byte/word select.

**Module Selection**—Two PROMs in this block monitor the address lines (BDAL 21-13) and compare it to the starting address jumper settings. In the -DC version of the module, the two PROMs are programmed to recognize four groups of 4K words. The starting address jumpers specify the first address of the four groups. If the address received belongs to one of the four groups, signal MSEL L is generated to force the module to be selected. The signals N/A14 and N/A13 are also generated to select the particular 4K group to which the address belongs.

In the -DA version of the module, the two PROMs in the block are programmed to recognize a particular 4K word group. (There are only 4K words on the module.) The starting address of this group is specified by jumper settings. If the received address belongs in this group, the PROMs enable MSEL L to be generated, forcing the module to be selected. The group selection signals, N/A14 and N/A13, will both be low.

**Address Latch**—These circuits store DAL 12-00, WTBT, N/A13, and N/A14 when the memory module has been selected and BSYNCH has been received. During an address cycle, signal WTBT specifies whether a write or a read is to be performed. WTBT asserted causes a write

cycle. WTBT deasserted causes a read cycle. During a data cycle, WTBT asserted specifies a write byte cycle, while WTBT deasserted causes a write word cycle.

The address latch also provides the following signals:

1. Address (A1-A11) to the memory array
2. Row selection control (N/A14, N/A13, and A12) to the row selection circuits
3. Address bit A0 to the byte/word select block

**Control and Timing**—This function decodes bus control signals received off the bus and generates internal timing and control signals. When the module has been selected and has received a DIN or DOUT signal, the control block makes a read or write request. This request starts module timing signals TIM 0 H, TIM 30, TIM 75, TIM 130, TIM 225, and TIM 275 H, that in turn, develop specific control signals. For example, TIM 0 enables row selection logic. Also, TIM 0 H and TIM 30 H develop the control signal OEL (output enable).

The control and timing circuits also prevent a memory access during power-down that could destroy data. During a power-down cycle, BDCOK H is deasserted, indicating that the power supply will shut down after 1 ms (minimum). In addition, all signals on the LSI-11 Bus will go to an unknown, uncontrolled state. BDCOK H causes an assertion of the LOCKOUT signal. LOCKOUT prevents an access to memory by locking rows 0 through 7 in the unasserted (high) state. LOCKOUT also initializes the control logic and switches the module to battery backup power.

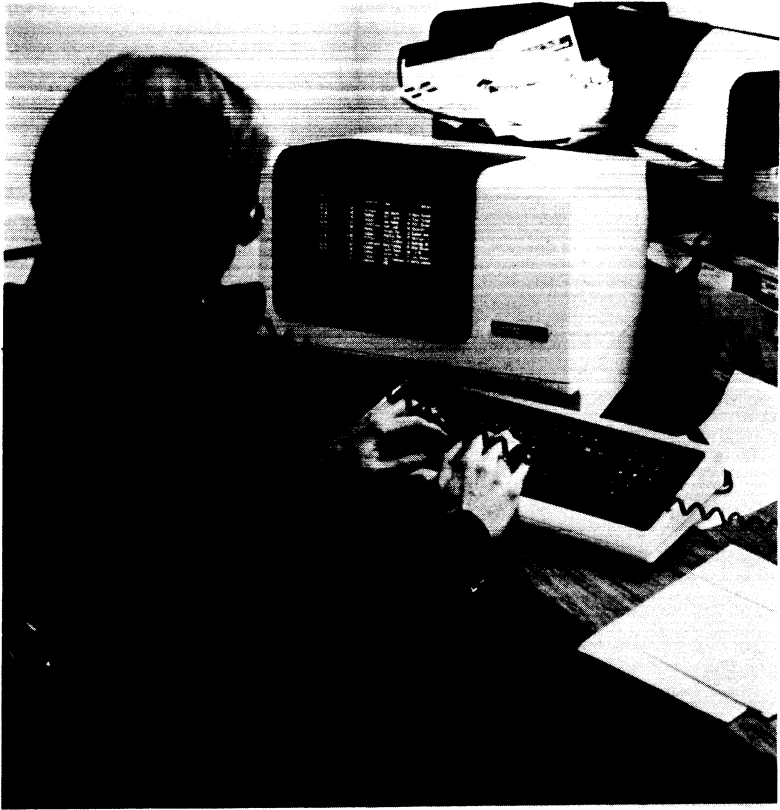
On a power-up cycle, the power supplies come into tolerance while BDCOK H is still deasserted. Within 4 ms of BDCOK H assertion, the LOCKOUT signal will deassert, initializing the logic and unlocking the array for memory access.

**Byte/Word Select**—If data is byte-formatted, this block determines if the high or low byte of a memory location is addressed. If address line A0 and signal WTBYTE are high, WTBYTE 1 L is asserted and the upper byte is selected. If A0 and WTBYTE are low, then WTBYTE 0 L is asserted and the lower byte selected. (If a word write is to be performed, WTBT goes low to assert both WTBYTE 1 L and WTBYTE 0 L.) Note that the byte not selected during a write byte operation will perform a dummy read.

**Memory Array**—Data is read from memory when the following conditions hold: OEL asserted, ROW asserted, WTBYTE 0 deasserted, and WTBYTE 1 deasserted. Data is written into memory when either or both of the WTBYTE signals are asserted and ROW are asserted.







**MULTIFUNCTION MODULES MXV11-A MEMORY AND ASYNCHRONOUS SERIAL LINE INTERFACE****INTRODUCTION**

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 32 KB (4K or 16K words). Two 24-pin sockets are provided for +5V 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 38,400 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 multilevel 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 — BENEFITS**

- 8K or 32K RAM — provides a program 'scratch pad'
- On-board RAM refresh — requires no external refresh logic.
- Accepts a variety of UVROMs, fusible-link PROMs, and a system device bootstrap — allows flexibility of memory functions.
- Two RS232 serial lines — provides industry-standard serial I/O.
- Jumper-selectable baud rate — provides flexibility of interface devices.

**Model Designations**

MXV11-AA	LSI-11 multifunction module, 8 KB Random Access Memory
MXV11-AC	LSI-11 multifunction module, 32 KB Random Access Memory

**SPECIFICATIONS**

Identification	M8047-AA	MXV11-AA	8 KB
	M8047-CA	MXV11-AC	32 KB
Size	Double		
Power	+5V, 1.2 A		
	+12V, 0.1 A		
Bus Loads			
AC	2		
DC	2		

**RAM Performance**

Bus Cycle Type	T <sub>acc</sub> (ns)	
	Typical	Max.
DATI	280	300
DATO(B)	395	410

**Notes:**

1. Access time (T<sub>acc</sub>) 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:	+5V ±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**

<b>UV PROMS</b>	<b>Chip Array Size</b>	<b>Memory Size</b>
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**

The user can configure the MXV11 features by using the jumpers provided on the board. The jumpers 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. Figure 20-1 illustrates the MXV11-A jumper locations. The soldered jumpers are factory-configured and should not be changed. When installing jumpers, the wire runs must be arranged so that no more than two wires are on each post and there is no level jumping between posts. Table 20-1 lists the factory-configured wiring scheme. Table 20-2 lists and describes the function of each jumper on the MXV11-A board.

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.

**Table 20-1 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	
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 (eight data bits, no parity, one stop bit)	J59	J61	
	J62	J64	
	J60	J63	
	J61	J62	
	J59	J66	
	J63	J65	
Baud Rates	CH0 (38,400)	J45	J50
	CH1 (9600)	J46	J48
Break Generation (Halt option)	J6	J7	
Crystal Clock	J68	J67	

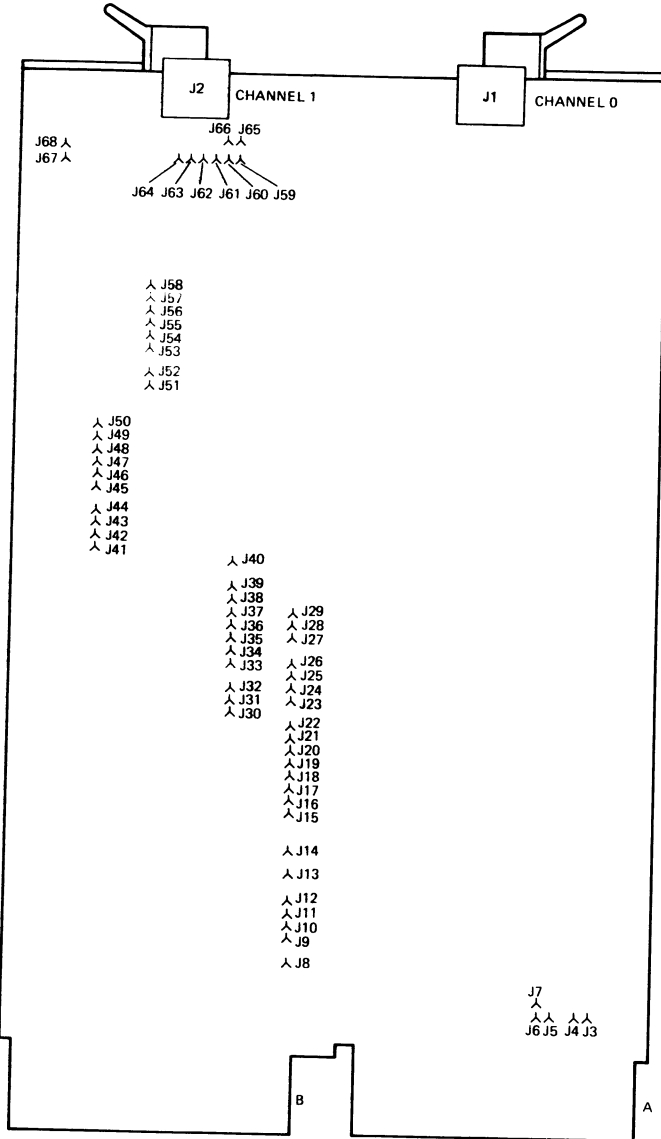


Figure 20-1 MXV11-A Jumper Locations

<b>Pin</b>	<b>Function</b>	<b>Option</b>
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 0 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 0 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 1 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 1.	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



<b>Pin</b>	<b>Function</b>	<b>Option</b>
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
J34	+3V. 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 +3V, 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 +3V, 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 +3V, to J35 for A12, to J36 for A11 or to J40 for +5V.	ROM
J40	+5V. Used to power some ROMs on pin 21.	ROM

<b>Pin</b>	<b>Function</b>	<b>Option</b>
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
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 19,200 baud.	SLU
J50	Used for 38,400 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

<b>Pin</b>	<b>Function</b>	<b>Option</b>
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
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	+3V. 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 +3V (J66) for eight bits with no parity.	SLU

<b>Pin</b>	<b>Function</b>	<b>Option</b>
J60	Two stop bits. Selects one or two stop bits for channel 1. Wirewrapped to ground (J65) for one stop bit or to +3V (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 (J65) for odd parity or to +3 V (J66) for even parity.	SLU
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. +3V 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

Pin	Function	Option
J68	Clock out. Crystal oscillator output at 19.6608 MHz. Wirewrapped to J67. Not a user option.	SLU

### Configuring the RAM

The RAM can be configured to start on any 8 KB boundary below 64 KB. Because of this restriction, the MXV11 (8 KB version) is not usable for memory above 56 KB. The MXV11 can be used in 18-bit memory address systems, but it is restricted to being assigned to the memory area below 56 KB.

Five wirewrap terminals, J30 through J34, select the starting address. Figure 20-2 shows the jumper configurations required to obtain the desired starting addresses.

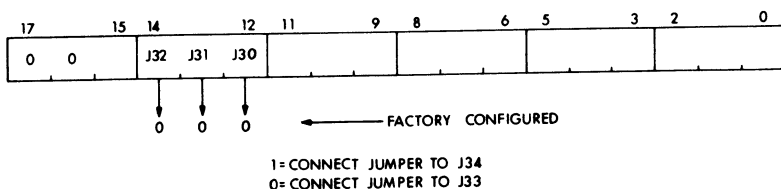


Figure 20-2 RAM Starting Address Selection

### Configuring the ROM

Depending on the ROM type, the module's capacity is 1K, 2K, or 4K words using a pair of  $1024 \times 8$ ,  $2048 \times 8$  or  $4096 \times 8$ -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 PROMs chapter, Chapter 18. 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 Figure 20-3 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.

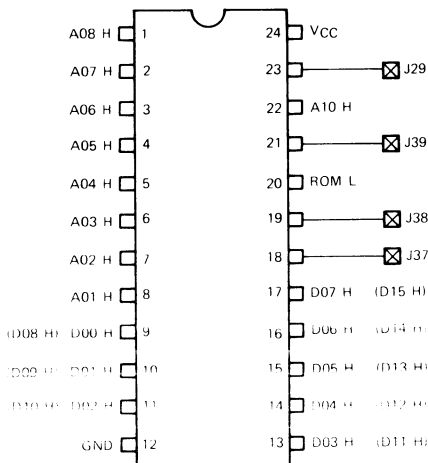
**ROM Bank Selection** — If the MXV11-A sockets are used for a program ROM instead of a bootstrap ROM, the memory must be selected by a jumper connecting J20 to J21. When main ROM memory is selected, the entire 4K-word bank is enabled. If a 1K or 2K ROM is used, it will “wrap around” and give invalid data depending on how the address lines are configured when the nonexisting ROM area is addressed. Main ROM memory may be positioned in bank 0 or bank 1 only. 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 Table 20-3.

**Configuring for Specific ROM Types** — Additional jumpers must be connected depending on the type of ROM used. Table 20-3 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.

**Table 20-3 EPROM Address Jumpers**

Function	From	2716 ROM		2732 ROM	
		Bank 0 To	Bank 1 To	Bank 0 To	Bank 1 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 20-3 MXV11-A ROM Socket Pin Assignment

### 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.

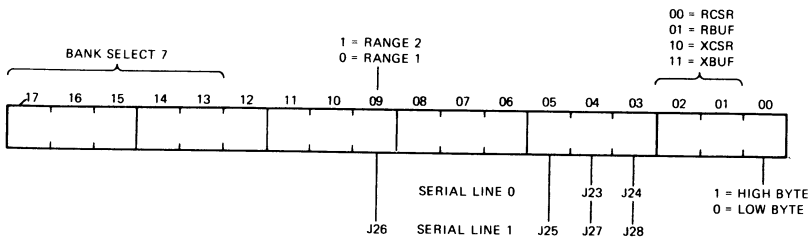
The format of an SLU address is shown in Figure 20-4. Note that bits 13:17 are neither configured on nor decoded by the MXV11-A module. These bits are decoded by the bus master module as the bank 7 select (BBS7 L) bus signal. This signal becomes active only when the I/O page is accessed. Bit 0 is used 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). Table 20-4 describes the jumper combinations to select one of four device addresses for serial line 0 (I/O).

**Table 20-4 Serial Line 0 Address Jumpers**

Address (Octal)	Jumper Posts	
	J23 to	J24 to
176500	J18 (Logic 0)	J19 (Logic 0) factory configura- tion
176510	J18 (Logic 0)	J12 (Logic 1)
176520	J13 (Logic 1)	J19 (Logic 0)
176530	J13 (Logic 1)	J12 (Logic 1)
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. Table 20-5 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 20-4 MXV11-A SLU Address Format**



**Table 20-5 Serial Line 1 Address Jumpers**

<b>Address (Octal)</b>	<b>Jumper Posts</b>			
	<b>J26 to</b>	<b>J25 to</b>	<b>J27 to</b>	<b>J28 to</b>
<b>Range 1</b>				
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
<b>Range 2</b>				
177560	J15	J14	J13	J19
(See note)				

**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 Registers**

The MXV11-A has two control/status registers (CSRs) for each of its two serial line units. Figure 20-5 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. Tables 20-6 through 20-9 describe the bit assignments for each register.

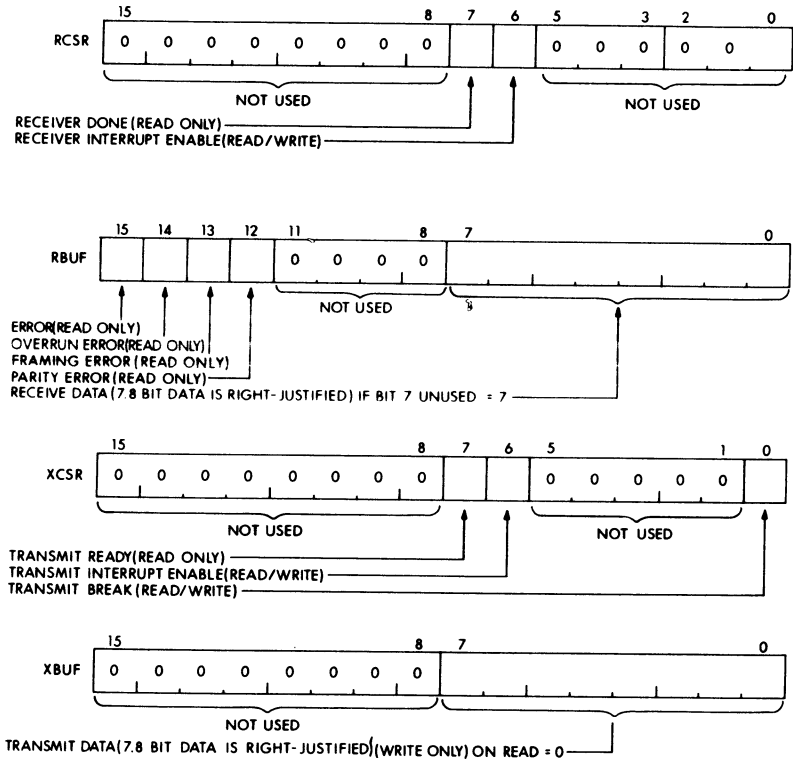


Figure 20-5 SLU CSR Formats

Table 20-6 Receiver Status Register

**Bit: 6 Name:** Interrupt Enable, read/write.  
**Function:** A 1 enables receiver interrupts, a 0 disables interrupts. Cleared by Initialize.

**Bit: 7 Name:** Receiver Done, read-only.  
**Function:** 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.

**Table 20-7 Receiver Data Register**

- Bit: 0:7 Name:** Data bits, read-only.  
**Function:** Bit 0 is the least significant bit, and bit 7 is the most significant. If seven data bits plus parity are selected, bit 7 will always read as a 0.
- Bit: 12 Name:** Parity Error, read-only.  
**Function:** 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 Name:** Framing Error, read-only.  
**Function:** 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 Name:** Overrun Error, read-only.  
**Function:** 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 Name:** Error, read-only.  
**Function:** A 1 indicates that one or more of bits 12, 13, and 14 are 1. Cleared when read or by Initialize.

**Table 20-8 Transmitter Status Register**

- Bit: 0 Name:** Break, read/write.  
**Function:** 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 Name:** Interrupt Enable, read/write.  
**Function:** A 1 enables transmitter interrupts, a 0 disables interrupts. Cleared by Initialize.
- Bit: 7 Name:** Transmitter Ready, read-only.  
**Function:** 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. Cleared when data are written into the transmitter data register. Set by Initialize.

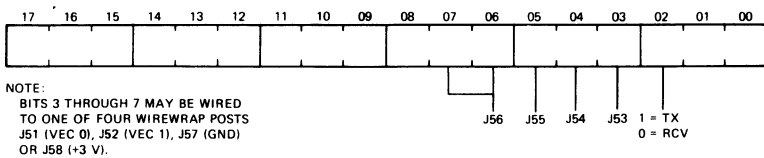
**Table 20-9 Transmitter Data Register**

**Bit:** 0:7    **Name:** Data bits, write-only.

**Function:** Bit 0 is the least significant bit and bit 7 is the most significant bit. If seven data bits plus parity are 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 transmit) are provided for each of the two serial lines. The interrupt vector format is shown in Figure 20-6. Each SLU port can be independently configured to operate in one of two ranges: 000 to 074, or 300 to 376. Table 20-10 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.



**Figure 20-6 Interrupt Vector Format**

**Table 20-10 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). Table 20-11 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).

**Table 20-11 SLU Vector Addresses Example**

Serial Line No.	Vector Base	Vector Address Bits				
		7	6	5	4	3
1 (Console)	060	0	0	1	1	0
0 (I/O)	310	1	1	0	0	1
<b>Jumpers</b>						
	<b>From</b>	J56	J56	J55	J54	J53
	<b>To</b>	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 odd or even parity or eight data bits without parity.

The parameters are selected by installing jumpers between the appropriate parameter post and J65 (Logic 0) or J66 (Logic 1). Table 20-12 describes the jumper configurations required for the desired serial line parameters.

**Table 20-12 Serial Line Parameter Jumpers**

SLU 0		SLU 1		Function
From Pin	To*	From Pin	To*	
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			**38,4K baud
		J46	J48	**9600 baud

\* Logic 1 = J66

Logic 0 = J65

\*\* Factory Configuration

### Bootstrap Jumpers

The MXV11-A2 is a pair of 24-pin ROM chips containing two bootstrap programs. Choosing which bootstrap to use is done with a wire-wrap jumper. To bootstrap from the TU58, wire the jumper from J29 to J15. To bootstrap from disk, wire the jumper from J29 to J15. To install the option, place the ROM marked 039D1 in socket XE57, and place 040D1 in socket XE67.

**MXV11 TU58 BOOTSTRAP** — The MXV11 TU58 bootstrap is a 256-word diagnostic and bootstrap program for LSI-11 systems using the TU58 DECTape II tape cartridge drives. The bootstrap performs the following functions:

1. On power up, sizes memory (up to 60 KB).
2. Tests addressing of memory by writing each location with its address and by verifying the contents.
3. Checks data retention, writing 1s into a 0s background, and vice-versa.
4. Attempts to bootstrap the TU58 cartridge on unit 0. If unit 0 fails to function, cycle to unit 1 and try it. Continue cycling between units 0 and 1 until one correctly reads block 0 of the tape.
5. Checks the first word read to see if it is 240<sub>8</sub>. If so, start program execution at location 0 with interrupts disabled (PR7). (This is the standard PDP-11 bootstrap convention.)

6. If the first word is not 240<sub>8</sub>, check to see if it is 260<sub>8</sub>. If so, this signals a special "stand-alone" program load request to this bootstrap program. If not 260<sub>8</sub>, return to step #4 and cycle to the next unit.
7. If a stand-alone program load request was made (i.e., location 0 contains 260<sub>8</sub>), scan the RT-11 directory of the unit to find the file whose name is given in locations 2-6 in block 0 of the cartridge. The name is represented as three words of RADIX50 data denoting the desired filename and execution.
8. If the file is found in the directory, load it as an RT-11 .SAV image file, and start its execution.

The bootstrap ROM also provides a MACRO-or FORTRAN-callable entry point which can be used to "chain" from one stand-alone program to another. The format of the FORTRAN CALL is:

CALL TULOAD( ifile)

where "ifile" is a four-element INTEGER array containing:

ifile(1) = the unit # of the TU58 drive to use (binary 0 or 1)

ifile(2) = the RADIX50 code for the first 3 characters of the filename

ifile(3) = the RADIX50 code for the last 3 characters of the filename

ifile(4) = the RADIX50 code for the file extension

Calling the entry point TULOAD with the appropriate FORTRAN-format argument list will cause the bootstrap to search the directory of the indicated tape unit for the file specified, and, if found, load it into memory and start its execution. The entry point for TULOAD is 173154.

**MXV11 DISK BOOTSTRAP** — This is a 256-word bootstrap program designed to handle all disks which are available for the LSI-11 Bus. It will automatically search for controllers for the various disks (in a predefined order) and bootstrap the first such device which is found and is operable.

The bootstrap sequence is as follows:

1. Size read/write memory, to a maximum of 60 KB. More memory may be present, but will not be tested or used by the bootstrap.
2. Perform a memory addressing test, writing each location with its address and verifying the contents.
3. Exercise memory data storage capabilities, moving a 1s pattern through a background of 0s and a 0s pattern through a background of 1s, to test all read/write memory locations for independence and retention.

4. Check for presence of RLV11 controller. If not found, proceed to Step #6.
5. Attempt to bootstrap RL01 or RL02 unit #0. If there is no such drive, or if no cartridge is present, the cover is open, or the drive is spinning down, proceed to Step #6.
6. Check for presence of RKV11 controller. If not found, proceed to Step #8.
7. Attempt to bootstrap each RK05 unit in sequence (0,1,...,7) until a unit is found which is ready and readable. If no such unit exists, proceed to Step #8. (This step is preceded by a minimum 8-second wait loop to allow sufficient spin-up time.)
8. Check for presence of RXV11 or RXV21 controller in the system. If none exists, proceed to Step #10.
9. Wait for minimum of 2 seconds to allow drive spin-up, then attempt to bootstrap unit 0 of the floppy disk, at the density of the media present in the drive at the time. If the drive is not ready or does not contain a bootable medium, go to the other unit and try it.
10. Check for the presence of an MRV11-C board with paging enabled. If not found, proceed to step #4.
11. Load the first 256 words from the MRV11-C into memory and execute them, if the first word is NOP (000240). Else proceed to step #4. Refer to Appendix G for Bootstrap HALT locations.

#### **NOTE**

The MXV11-A2 boot ROM and the .116 BDV11 boot ROM do not initialize memory locations above 30K words. The user must supply a program to initialize those locations or parity errors will occur. To initialize a location, read from it and then write to it.

#### **BAUD RATE JUMPERS**

Each serial line can be configured for internal baud rates from 150 to 38,400 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 20-13. 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-A should be jumpered for an external baud rate.

**Table 20-13 Baud Rate Jumpers**

<b>From</b>	<b>To</b>	<b>Function</b>
J45		SLU0
J46		SLU1
	J41	150 baud clock
	J43	300 baud clock
	J42	1200 baud clock
	J44	2400 baud clock
	J47	4800 baud clock
	J48	9600 baud clock
	J49	19,200 baud clock
	J50	38,400 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 Table 20-14.

**Table 20-14 Serial Line 1 Break Response Jumpers**

Break Response, Operation	Jumper Posts	
	From	To
Reboot	J6	J5
Halt (factory configuration)	J6	J7
No Response	No Jumper installed	

**60 Hz Clock** — A 60 Hz clock is derived from the crystal on the MXV11-A. It can be jumpered to the BEVNT line to provide the equivalent of a line time clock for a system which does not otherwise have one. In the factory configuration, this signal is 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-A 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-A 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

Table 20-15 lists 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. 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.

**Table 20-15 Definition of Cables**

<b>Cable</b>	<b>Application</b>	<b>Length</b>
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 (19,200 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 20-7 and 20-8 show the variety of cables and options used with the MXV11-A as well as the primary application of each.

When designing a cable for the DLV11-J, several points to consider are:

1. 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. (See Figure 20-9)
2. The receivers on the MXV11-A 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 Figure 20-10)

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

### 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 20-8 shows the cables and devices which may be used with the DLV11-KA option.

### DESCRIPTION

The memory and serial line block diagram is illustrated in Figure 20-12. 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.

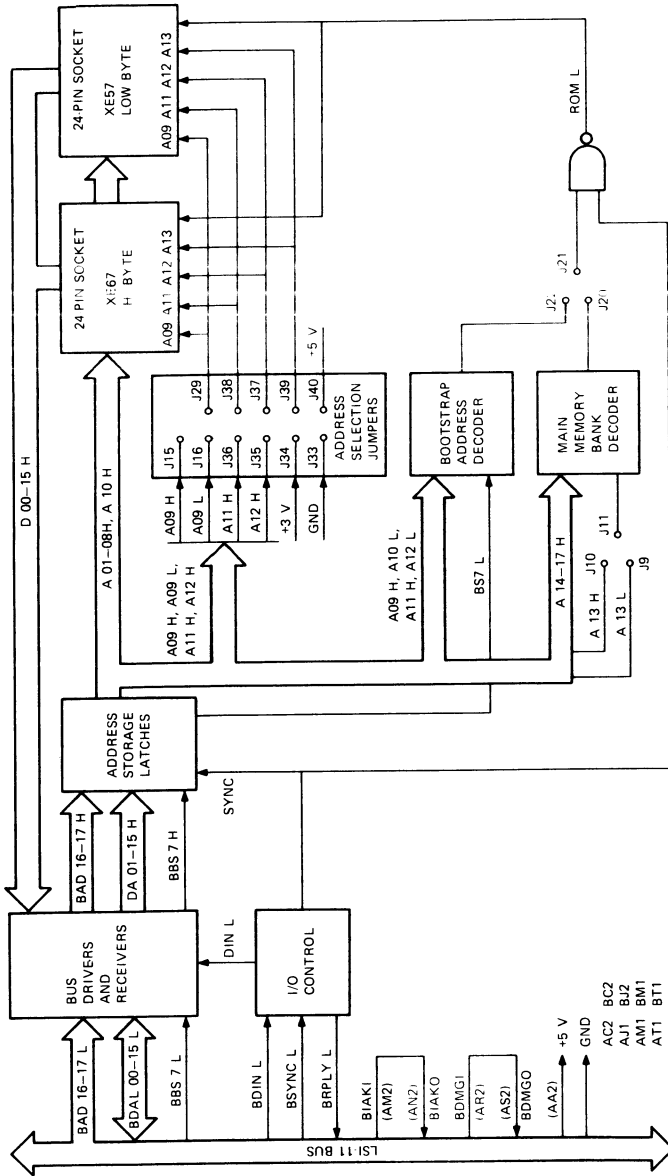


Figure 20-13 MXV11-A RAM Block Diagram

**Memory Read** — When in a memory read operation, the bus master device asserts BDIN L. The data from the accessed memory location are 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 constituting 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 constituting the low (even) byte of the memory. The reply logic also responds to the active BDOUT L signal by asserting BRPLY L, indicating that data have 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. Figure 20-14 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 (38,400, 19,200, 9600, 4800, 2400, 1200, 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 Circuit** — The charge pump circuit is a dc-to-dc converter which provides the  $-12\text{V}$  power for the serial communications driver and  $-5\text{V}$  for the MOS memory array integrated circuits. The input power for this circuit is  $+12\text{V}$ . A 307.2K Hz signal obtained from the baud rate generator is applied to a rectifier circuit which produces a  $-12\text{V}$  output. The  $-12\text{V}$  output is applied to a resistor and zener diode to produce the  $-5\text{V}$  output. Note that this circuit eliminates the need for backplane power other than the standard  $+5\text{V}$  and  $+12\text{V}$ .

### **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. Figure 20-14 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.

**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  $+3\text{V}$  (J34),  $+5\text{V}$  (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

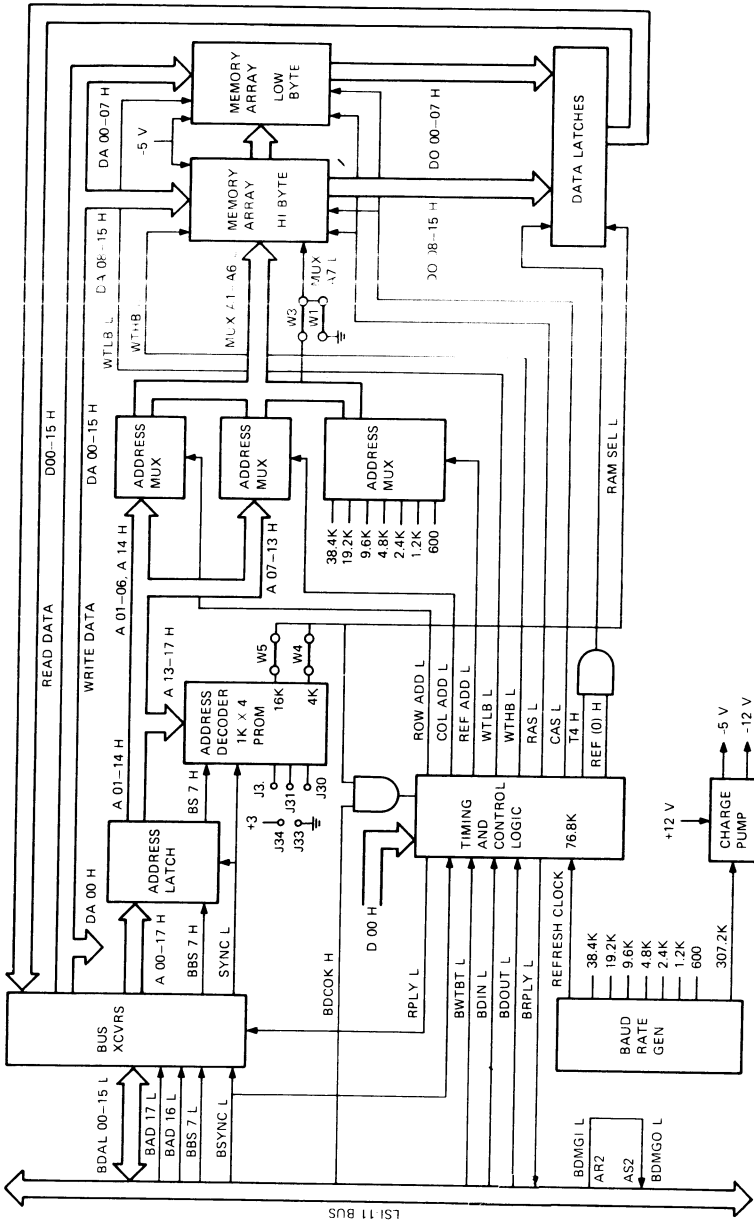


Figure 20-14 MXV11-A ROM Block Diagram



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 are available within 450 ns after the BSYNC L signal is received by the MXV11-A. ROM logic on the module responds only to DAT1 and DA-TIO(B) bus cycles. ROM logic functions are not affected by the bus initialize (BINIT L) signal.

### **Serial Line Interface**

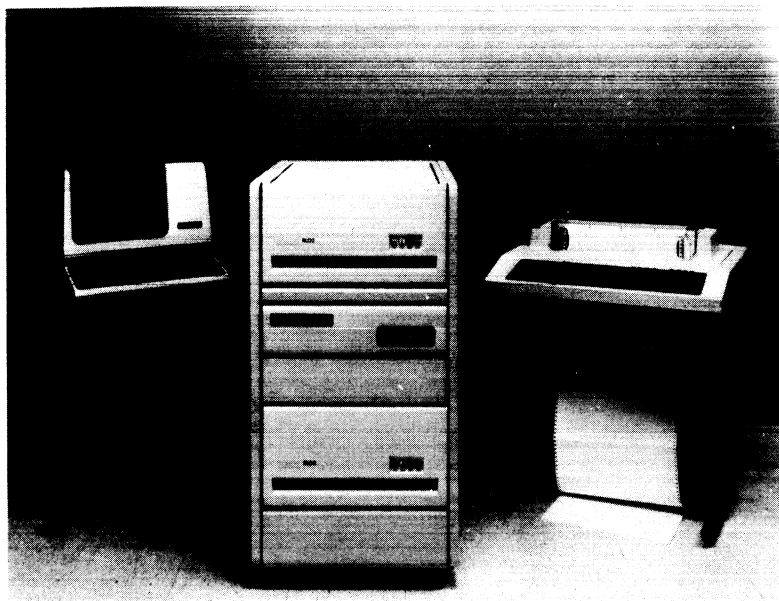
Data pass through four main circuits when moving to and from a peripheral device. The serial transmitter data leave the SLU and enter the EIA transmitter driver where the data are converted from TTL to EIA-compatible bipolar levels. The data leave the EIA converter on an interface cable and enter the 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, 1200, 2400, 4800, 9600, 19,200 and 38,400 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.

**section III**  
**systems**





## CHAPTER 21

# SYSTEMS

DIGITAL's semiconductor developments have led to extending the PDP-11 microcomputer systems family to include the low-cost PDP-11/23-PLUS, PDP-11/23, and PDP-11/03 microcomputers.

An LSI-11 microcomputer board packaged with a backplane, power supply, and rack-mountable box is considered a member of DIGITAL's 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-L. The LSI-11/23 boxed with a backplane, memory, and power supply, becomes a PDP-11/23. These low-end members of the PDP-11 family have most of the PDP-11 family features and benefits mentioned in the first chapter of the Commonalities section of this Handbook.

### Levels of Integration

Like all DIGITAL's PDP-11 family, the PDP-11/23-PLUS, PDP-11/23, and PDP-11/03 processors are available at two levels of integration:

- Fully configured systems with mass storage and a choice of system software. The full range of LSI-11 Bus options is available to provide functionality appropriate to the specific application.
- Box-level processors are assembled and tested units with the CPU, memory, and, in the PDP-11/03-L and the PDP-11/23, bootstrap and diagnostic ROM with at least one serial line unit. LSI-11 Bus options and peripherals, together with DIGITAL cabinets and other accessories, allow maximum configuration flexibility.

The PDP-11/23-PLUS is not sold at the module level. However, extended addressing is available to component customers by using the LSI-11/23 CPU module in conjunction with the H9275 backplane, MSV11-L memory, and RLV12 disk controller. The PDP-11/23-PLUS CPU, the KDF11-B, is discussed in detail in the PDP-11/23-PLUS chapter, Chapter 22.

The CPUs used in the PDP-11/23 and PDP-11/03 processors are available at the module level, as described in this Handbook:

- The PDP-11/23 system uses the LSI-11/23 processor.
- The PDP-11/03-L system uses the LSI-11/2 processor.
- The PDP-11/03 system uses either the LSI-11 or the LSI-11/2 processor.

## FUNCTIONALITY AND PERFORMANCE

### **PDP-11/23-PLUS Systems**

The PDP-11/23-PLUS systems and box layered products, announced in November 1981, are positioned between the PDP-11/23 and the PDP-11/24 in price, functionality, and expandability. The PDP-11/23-PLUS is offered as the lowest priced packaged system from DIGITAL that provides full functionality using DIGITAL software RSX-11M-PLUS, RSTS/E, or CTS-500. This processor system can also address up to one full megabyte of parity MOS MSV11-P memory. The new MSV11-P memories provide easier isolation of memory faults. The PDP-11/23-PLUS system is also designed with sufficient expansion slots so that most applications will fit in a single CPU box. This eliminates the need and expense of an expander box.

The Commercial Instruction Set (CIS) is implemented by the PDP-11/23-PLUS and is standard in commercial systems. The PDP-11/23-PLUS has been designed with a system distribution panel that makes system installation and relocation easier.

There are two versions of the PDP-11/23-PLUS processor- the PDP-11/23-BC and the PDP-11/23-BE. The PDP-11/23-BC utilizes 256 KB of MSV11-PK MOS memory, while the PDP-11/23-BE utilizes 512 KB of MSV11-PL MOS memory. The PDP-11/23-PLUS box-level processors are supplied in a 5.25 inch standard rack-mounted box with a CPU which uses the same microprocessor chip set as the PDP-11/23, PDP-11/24, and the LSI-11/23. The PDP-11/23-PLUS CPU multifunction quad board enables easier troubleshooting. PDP-11/23-PLUS system products also contain diagnostic and bootstrap ROM on the CPU module, a line-time clock, parity memory, two asynchronous serial lines, a switchable 120/240 volt power supply set, and seven quad-height Extended LSI-11 Bus slots for options. Please refer to the example of a typical PDP-11/23-PLUS processor, below.

### **PDP-11/23 Systems**

The PDP-11/23, introduced in 1979, provides LSI-11 Bus users all of the CPU functionality, and most of the performance of DIGITAL's PDP-11/34, up to a maximum of 256 KB memory addressing. This makes PDP-11 memory management available to a whole new class of users, giving them the capability to run multiuser, multiprogramming operating systems such as RSX-11M. The BDV11 bootstrap/diagnostic ROM module tests the PDP-11/23 on power-up or restart, and will support system software loading from DIGITAL system devices or from a DECnet communication line. A DLV11-J module supports the console terminal as well as three additional asynchronous devices.

11/23-BC 256KB  
 -BE 512KB

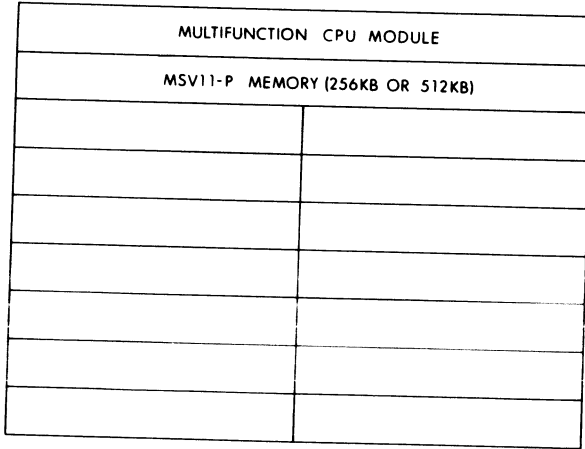
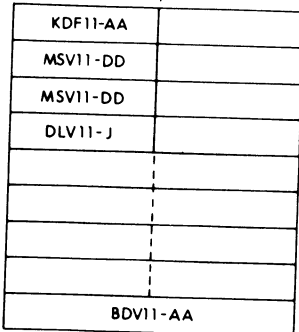


Figure 21-1 PDP-11/23-PLUS Rack Mountable Processor

Figure 21-2a illustrates the PDP-11/23 processor configuration utilizing 128 KB MOS memory. Figure 21-2b illustrates the PDP-11/23 processor utilizing 256 KB MOS memory.

11/23-AA (120 VAC)  
 -AB (240 VAC)



- LSI-11/23 MICROPROCESSOR
- 128KB MOS MEMORY
- EIS STANDARD
- BOOT/DIAGNOSTIC ROM STANDARD
- 4 ASYNC SERIAL LINES
- 4 QUAD OR DOUBLE HEIGHT SLOTS

Figure 21-2a PDP-11/23 Processor with 128 KB MOS Memory

11/23 - AC (120 VAC)  
- AD (240 VAC)

KDF11-AA	
MSV11-DD	
MSV11-DD	
MSV11-DD	
MSV11-DD	
DLV11-J	
BDV11-AA	

- LSI-11/23 MICROPROCESSOR
- 256KB MOS MEMORY
- EIS STANDARD
- BOOT/DIAGNOSTIC ROM STANDARD
- 4 ASYNC SERIAL LINES
- 2 QUAD OR DOUBLE HEIGHT SLOTS

Figure 21-2b PDP-11/23 Processor with 256 KB MOS Memory

### PDP-11/03-L Systems

The PDP-11/03-L is the 'large box' PDP-11/03 system. It is similar in appearance and features to the PDP-11/23, supports memory addressing to 64 KB, and provides cost savings for users who do not require memory management or the performance advantages of the PDP-11/23. Figure 21-3a illustrates the PDP-11/03-L processor configuration utilizing 32 KB MOS memory. Figure 21-3b illustrates the PDP-11/03-L processor configuration utilizing 64 KB MOS memory.

11/03 - LH (120 VAC)  
- LJ (240 VAC)

KD11-HA & KEV11	
MSV11-DC	
BDV11 - A A	

- LSI-11/2 MICROPROCESSOR
- 32KB MOS MEMORY
- EIS/FIS(KEV11) STANDARD
- BOOT/DIAGNOSTIC ROM STANDARD
- 6 QUAD OR 16 DOUBLE HEIGHT SLOTS

Figure 21-3a PDP-11/03-L Processor with 32 KB MOS Memory





Figures 21-5a - 21-5d illustrate various pictorial views of the PDP-11/03-L processor.

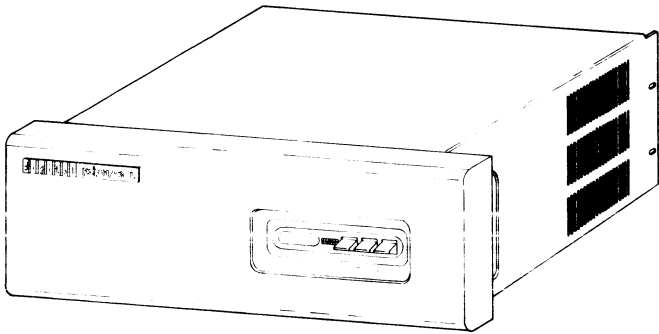


Figure 21-5a PDP-11/03-L

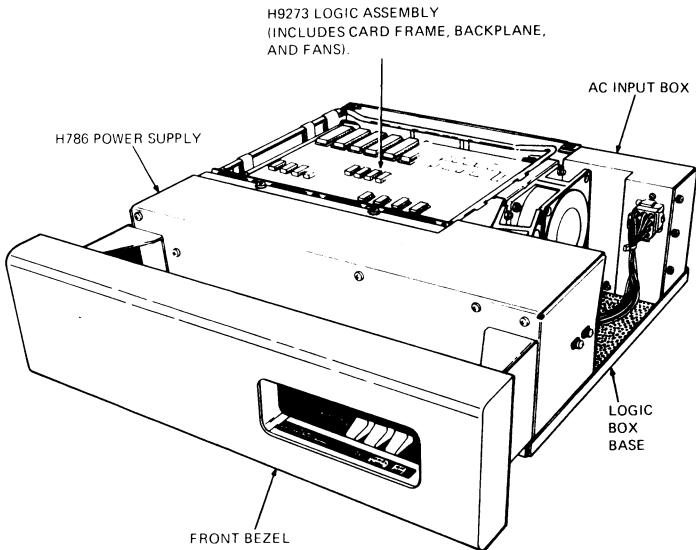


Figure 21-5b PDP-11/03-L with Cover Removed

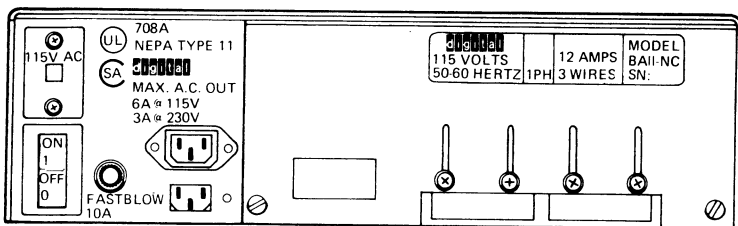


Figure 21-5c PDP-11/03-L Rear View

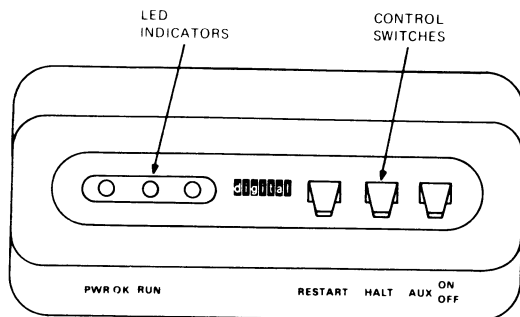


Figure 21-5d PDP-11/03-L Front Panel Switches and Indicators

### PDP-11/23-PLUS PACKAGED SYSTEMS

There are two categories of PDP-11/23-PLUS packaged systems—Technical packaged systems and Commercial packaged systems.

- Technical packaged systems consist of the PDP-11/23-PLUS with either 256 KB or 512 KB of parity memory, two 10 MB RLO2 disk drives, two serial line units in a 40 inch H9642 cabinet, six quad-height Extended LSI-11 Bus slots available for options, a choice of software that includes either RSX-11M-PLUS (512 KB) RSX-11M or RSTS/E (256 KB), and a VT100 or LA120 console terminal. Figure 21-6 depicts an example of a technical packaged system.

MULTIFUNCTION CPU MODULE	
MSV11-P MEMORY (256KB OR 512KB)	
RLV12 RLO2 DISK CONTROLLER	

Figure 21-6 Technical Packaged System

- Commercial packaged systems include the PDP-11/23-PLUS CPU with either 256 KB or 512 KB of parity memory, two 10 MB RLO2 disk drives, two DLV11-type serial lines and four multiplexed serial lines, five quad-height Extended LSI-11 Bus slots available for options, a choice of software that includes either RSX-11M-PLUS (512 KB) or RSX-11M, RSTS/E, CTS-500 (256 KB), and a VT100 or LA120 console terminal. Figure 21-7 illustrates an example of a commercial packaged system. Figures 21-8a - 21-8c illustrate various pictorial views of the PDP-11/23-PLUS processor.

MULTIFUNCTION CPU MODULE	
MSV11-P MEMORY (256KB OR 512KB)	
DZV11 4-LINE ASYNCHRONOUS MULTIPLEXER	
RLV12 RL02 DISK CONTROLLER	

Figure 21-7 Commercial Packaged System

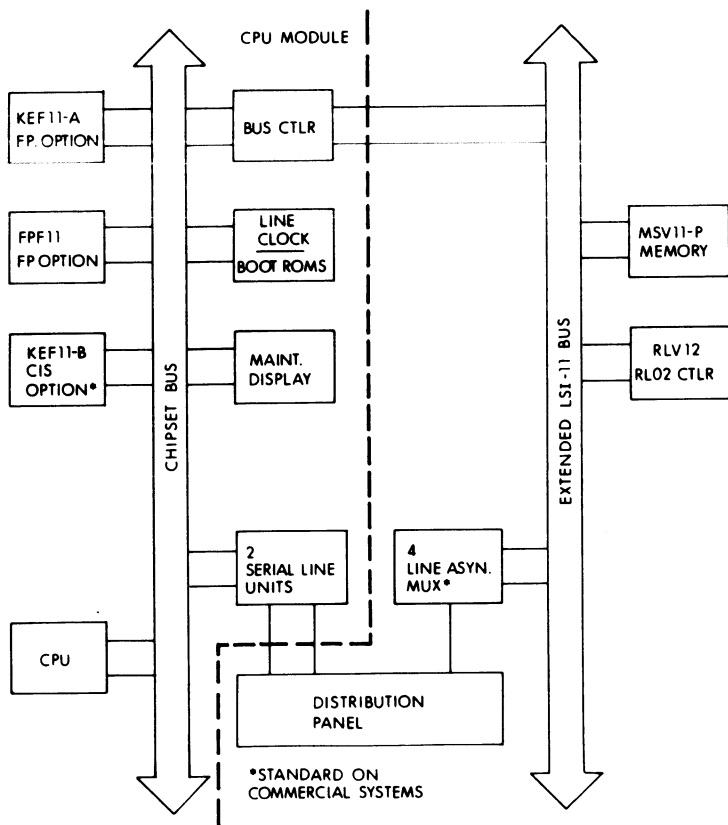


Figure 21-8a PDP-11/23-PLUS Hardware Organization

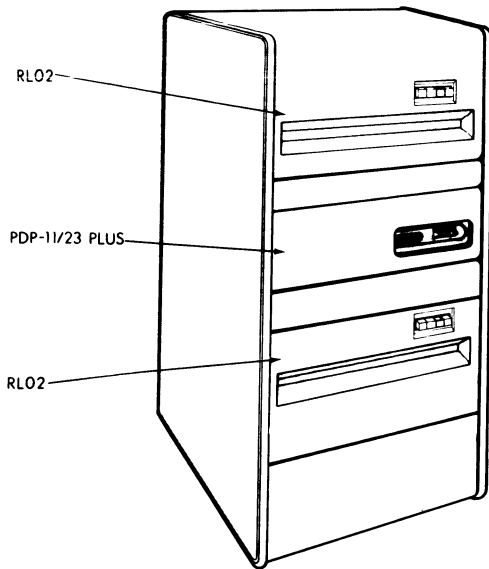


Figure 21-8b PDP-11/23-PLUS Front View

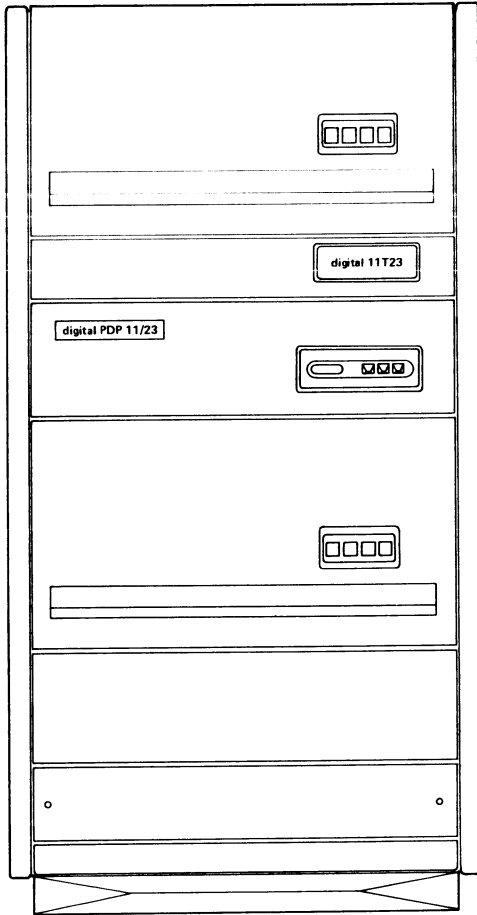


Figure 21-9c PDP-11T23 Computer System

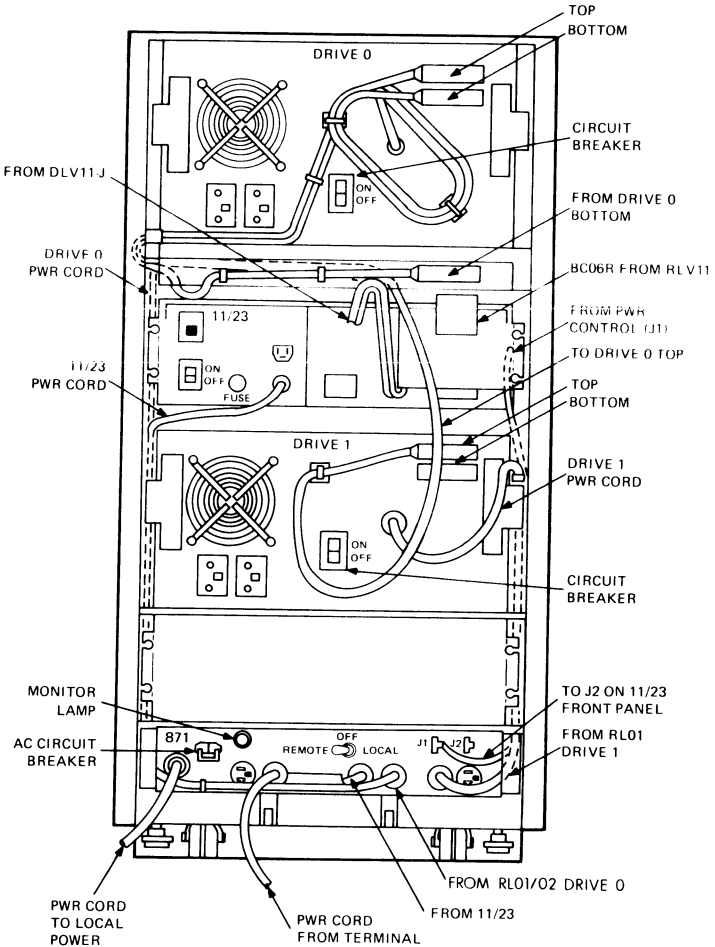


Figure 21-9d PDP-11T23 with Rear Panel Removed

- PDP-11V23 and PDP-11V03-L systems, which are floppy disk based, include a dual-drive RX02 floppy disk subsystem (1 Mb total) in a 30-inch tall H9610 cabinet. Figure 21-10a illustrates the PDP-11V23 configuration. Figure 21-10b illustrates the PDP-11V03-L configuration. Figures 21-10c and 21-10d illustrate various pictorial views of the PDP-11V23 computer system.



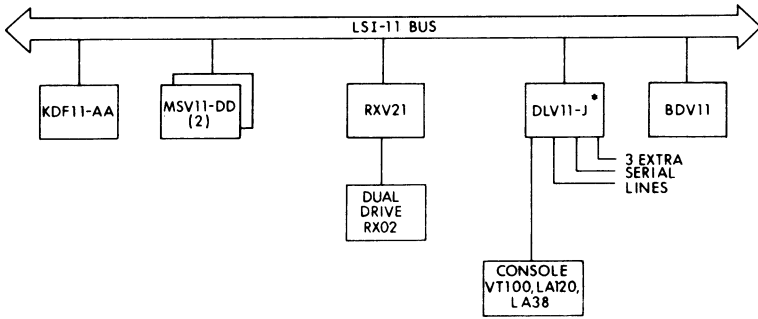
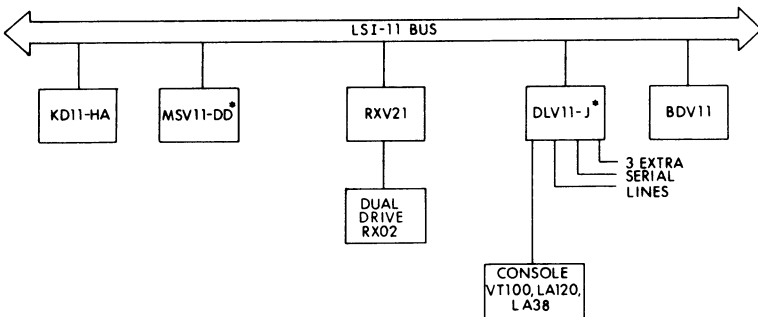


Figure 21-10a PDP-11V23 Configuration



\* IN 32KB 11V03L SYSTEMS, MSV11-DC REPLACES MSV11-DD  
AND  
DLV11-F REPLACES DLV11-J

Figure 21-10b PDP-11V03-L Configuration

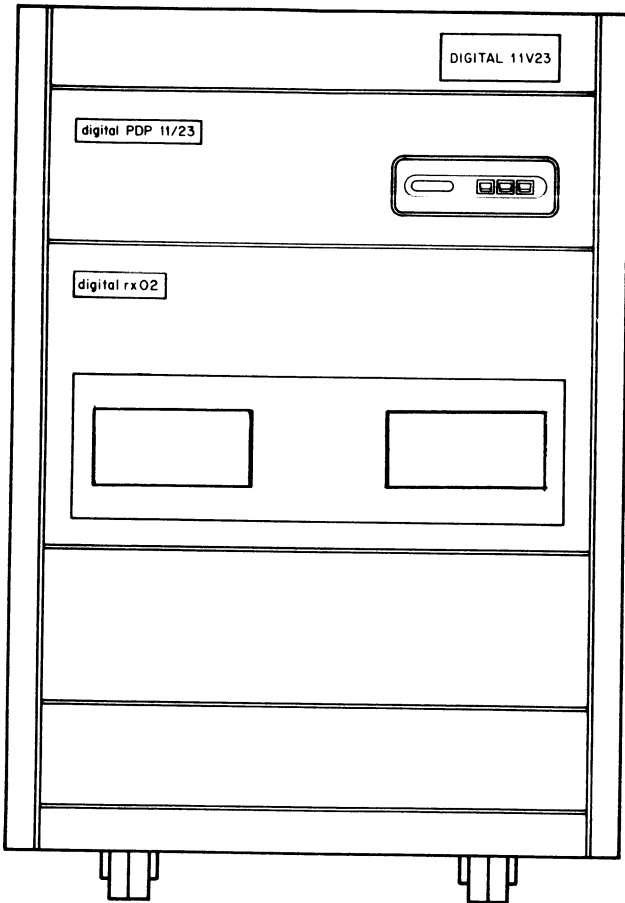


Figure 21-10c PDP-11V23 Computer System

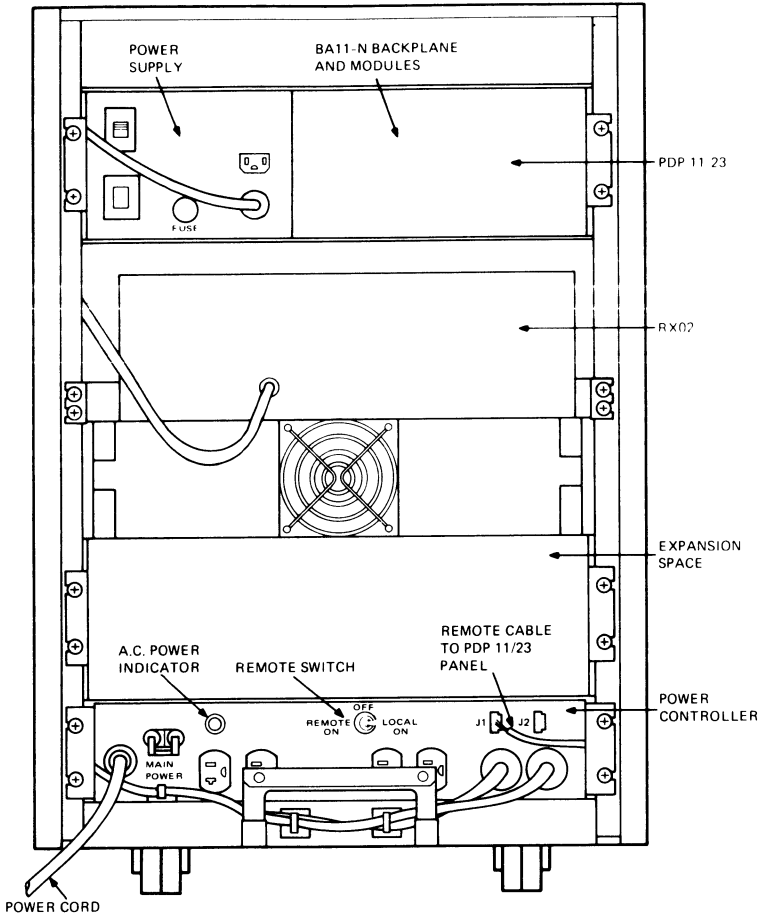


Figure 21-10d PDP-11V23 with Back panel Removed

The product designations for these systems are based on a nine-digit code (the 2-5-2 number) which describes the processor, amount of memory, mass storage devices, system software, console terminal and line voltage, as described below. In each case, the -XX is replaced by a code to designate the console terminal (VT100, LA120, LA38) and line voltage, and software support level.

**PDP-11/23-PLUS:**

SN-RXMMC-XX	PDP-11/23-PLUS based systems with RSX-11M-PLUS, 512 KB of memory, and two RL02 disk drives
SM-RXMMB-XX	PDP-11/23-PLUS based systems with RSX-11M, 256 KB of memory, and two RL02 disk drives
SE-RXMMB-XX	PDP-11/23-PLUS based systems with RSTS/E, 256 KB of memory, and two RL02 disk drives

**PDP-11T23:**

SM-WXMMA-XX SM-WXLLA-XX	PDP-11/23 based systems with RSX-11M, 128 KB of memory, and two RL02 (MM) or RL01 (LL) disks
SR-WXMMA-XX SR-WXLLA-XX	A PDP-11/23 based system with RT-11, 128 KB of memory, and two RL02 (MM) or RL01 (LL) disks

**PDP-11V23:**

SR-WXSSA-XX	A PDP-11/23 based system with RT-11, 128 KB of memory, and dual RX02 floppy disk mass storage
-------------	---

**PDP-11T03-L:**

SR-VXLLB-XX	A PDP-11/03-L based system with RT-11, 64 KB of memory, and two RL01 disks
-------------	--

**PDP-11V03-L:**

SR-VXSSB-XX	A PDP-11/03-L based system supporting RT-11 with 64 KB of memory, and dual RX02 floppy disk mass storage
SR-VXSSA-XX	A PDP-11/03-L, RT-11, dual RX02 system with 32 KB of memory

**SYSTEM EXPANSION**

PDP-11/23-PLUS systems have five slot positions available for adding options. Adding a BA11-S expander box provides seven additional slot positions for a total of twelve slots for system expansion. The expander box is available in two versions: the BA11-SC for 120 Vac systems, and the BA11-SD for 240 Vac systems. The 120 Volt systems must have a 874-C power controller if an expander box is installed. In

addition to the expander box, the BC02D-03 jumper/cable terminator assembly is required to connect the existing backplane to the backplane in the expander box. Figure 21-11, below, illustrates PDP-11/23-PLUS system expansion capabilities.

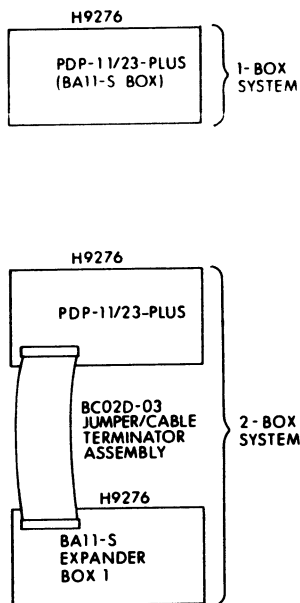


Figure 21-11 PDP-11/23-PLUS System Expansion

The PDP-11/23 systems are designed to support expansion to an additional BA11-N processor box for applications that require more room for expansion than that provided in the basic PDP-11/23 box. The 120 volt systems are supplied with a 20 amp plug to provide this capability.

PDP-11/03-L systems have the physical space for an expander box. However, 120 Vac PDP-11/03-L systems are provided with a 15 amp plug for easiest installation, which precludes expansion within UL ratings. A 120 Vac PDP-11/03-L system that requires expansion requires replacement of the standard 871-A 12 amp power controller with an

871-C 16 amp power controller, or installation of a second cabinet with power controller. Figure 21-12 illustrates PDP-11/23, PDP-11/03, and PDP-11/03-L system expansion capabilities.

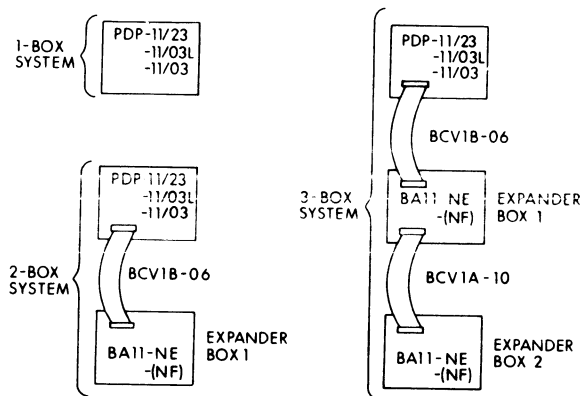


Figure 21-12 PDP-11/23, PDP-11/03-L, PDP-11/03 System Expansion

### PROCESSOR SELF-DIAGNOSIS

Each PDP-11/23-PLUS, PDP-11/23, and PDP-11/03-L processor includes bootstrap/diagnostic ROMs. In the PDP-11/23 and the PDP-11/03-L, these ROMs are contained on the BDV11 module. For the PDP-11/23-PLUS, this functionality is contained on the CPU module. Each time the system is powered up or restarted, the diagnostic programs in read-only memory will (as selected by the user):

- Test the CPU
- Test memory
- Test the console device and initiate a dialogue
- Indicate its status on a set of five light-emitting diodes visible from the back of the system
- Specify any error conditions with a six-digit error code at the console terminal

This capability increases user confidence in the system, and simplifies diagnosis and repair when necessary.



**PDP-11/23-PLUS MICROCOMPUTER****INTRODUCTION**

DIGITAL's compact PDP-11/23-PLUS microcomputer system, positioned between the PDP-11/23 and the PDP-11/24 in price, functionality, and expandability, is designed to increase system performance and provide efficient backplane utilization. This low-cost, 16-bit microcomputer can address up to one full megabyte of parity MOS MSV11-P memory while offering full PDP-11 processor functionality. It is also compatible with PDP-11 family software and hardware design, using the Extended LSI-11 Bus. The PDP-11/23-PLUS uses the same microprocessor chip set as the PDP-11/23, PDP-11/23, and the LSI-11/23.

The PDP-11/23-PLUS CPU module contains diagnostic and bootstrap ROM, a memory management unit, line-time clock, two asynchronous serial lines, and three sockets for the Commercial Instruction Set (CIS) and Floating Point Instruction Set options. Because of this unique packaging density, most applications will fit in a single PDP-11/23-PLUS CPU box. This eliminates the need and expense of an expander box.

The PDP-11/23-PLUS has been designed with a system distribution panel that simplifies access, installation, reconfiguration, and relocation of all serial line and option connections.

**FEATURES — BENEFITS**

- 91 single-and-double operand instructions — operate with bit, byte, 16-bit word, and multiple-word data types.
- 12 addressing modes — extend the standard instruction set to over 400 powerful instructions.
- Extended Addressing, up to one megabyte — supports more users per system and provides increased system performance.
- Optional Floating Point instruction set — increases FORTRAN and BASIC execution speeds.
- Optional Commercial Instruction Set — enables faster COBOL execution.
- Extended LSI-11 Bus interface — supports a full megabyte of memory.
- System distribution panel — for easier system installation, reconfiguring, and relocation.
- Expansion capability — provided by quad-height option slots to eliminate need for an expander box.



- Line frequency clock — provides the system with timing information at fixed intervals.
- 4 level vectored interrupts — allow a higher-level interrupt request to interrupt a lower-priority service routine and eliminate the need for interrupt polling.
- Stack processing — simplifies the handling of structured data, sub-routines, and interrupts.
- Direct Memory Access (DMA) — allows peripherals to access memory without requiring processor intervention.
- Power-Fail/Automatic restart hardware — allows the CPU to react to a power failure and resume operation upon power restoration.
- Console Emulator ODT — for debugging programs.
- Memory management — allows relocation and protection needed in multitask environments.
- Two DLV11 type serial line units — for the system console, and an additional terminal, line printer, or TU58 cartridge tape.
- Diagnostic/Bootstrap ROM — performs CPU, memory, and I/O device tests on power-up to ensure data integrity.

## SPECIFICATIONS

Identification	M8189
Size	Quad
Dimensions	CPU chassis is 13.2 cm high × 48.3 cm wide × 68 cm deep (5.2 in × 19 in × 26.8 in)
Power Consumption	+5 V ± 5%, at 6.4 A max. (at 4.5 A typ) +12 V ± 5%, at .7 A max. (at .3 A typ)
Bus Loads	2 AC unit loads 1 DC unit load
Nonoperating Environment	−40°C to 66°C (−40°F to 151°F), 10% to 90% relative humidity, noncondensing
Operating Environment	5°C to 50°C (41°F to 122°F), 10% to 95% relative humidity Maximum outlet temperature rise of 5°C (9°F) above 60°C (140°C)
Altitude	To 15.25 km (50,000 feet for Nonoperating Environment. To 2.44 km (8000 feet) for Operating Environment
Instruction Timing	Based on 75 ns intervals (See Appendix J.)

Interrupt Latency	5.7 microseconds (typical) 12.6 microseconds, max. (except EIS) 54.225 microseconds, max. (including EIS)
Interrupt Service Time	8.625 microseconds (memory management off) 9.750 microseconds (memory management on)
DMA Latency	1.35 microseconds, max.

**NOTE**

Interrupt and DMA latencies assume a KDF11-B with memory management enabled and using MSV11-P Memory.

**DESCRIPTION**

**Central Processor**

The PDP-11/23-PLUS central processor unit is contained on two LSI chips, control and data, which reside on a single 40-pin carrier (a dual in-line package). The standard Memory Management Unit (MMU) is contained on one LSI chip which also resides on a 40-pin carrier. PDP-11/23-PLUS contains sockets for these two carriers, plus three extra sockets which are reserved for the Commercial Instruction Set (CIS) or floating point options.

The architecture of the PDP-11/23-PLUS is highly expandable because of its internal bus structure. The control and data chips communicate with each other, as well as with the external PDP-11/23-PLUS logic, over the Micro Instruction Bus (MIB) <15:00> and Chip Data and Address Line (CDAL) <21:00> bus. PDP-11/23-PLUS logic interfaces these chips to the Internal Data and Address Line (IDAL) <15:00> bus and the external Extended LSI-11 Bus. The IDAL bus provides additional loading capacity on the chip set bus. For an illustration of the positions of these busses, please refer to Figure 22-1, the PDP-11/23-PLUS functional block diagram.

The PDP-11/23-PLUS boot and diagnostic ROMs, line clock, and serial line units reside on the IDAL bus. Memory and additional peripherals interface to the Extended LSI-11 Bus. Bidirectional interfaces (CDAL/IDAL) transceivers and CDAL/BDAL transceivers on the PDP-11/23-PLUS CPU module connect the CDAL <21:00> bus with the IDAL <15:00> bus. PDP-11/23-PLUS logic supporting LSI chip set includes the master clock control logic, MIB decode logic, fixed data logic, service logic, reset logic, and ODT logic. Logic pertaining to the Extended LSI-11 Bus includes the bus control logic, bus synchronizer,



and the CDAL/BDAL transceivers. Logic pertaining to the IDAL bus peripherals includes the bus control logic, CDAL/IDAL transceivers, the IDAL address decode, the boot/diagnostic ROMs, the line clock logic, the console and second SLU logic, the baud rate generator, and the -12V charge pump circuits.

### **Floating Point Option**

Forty-six floating point instructions are available as a microcode option (KEF11-A) on the PDP-11/23-PLUS processor to supplement the integer arithmetic instructions in the basic instruction set. The Floating Point Instruction Set executes floating point operations much faster than equivalent software routines. It also provides both single-precision (32-bit) and double-precision (64-bit) operands, and conserves memory space by executing in microcode instead of software. This option implements the same Floating Point Instruction Set available with DIGITAL's PDP-11/34, PDP-11/60, and the PDP-11/70 minicomputers.

### **Commercial Instruction Set**

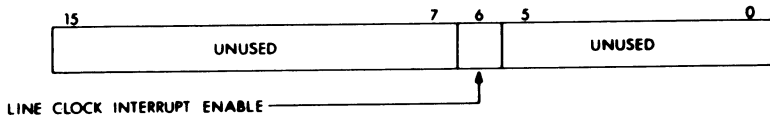
The Commercial Instruction Set (CIS), a microcode option, adds character string instructions to the basic PDP-11 instruction set. These character string operations implement functions of commercial data and text processing applications. CIS microcode resides in six MOS/LSI chips which are mounted on a single double-width 40-pin carrier.

### **Line Clock**

Line frequency clock provides the system with timing information at fixed intervals. The intervals are synchronized with the line frequency of the user's input power. The line clock generates bus request level 6 interrupts to the processor at time intervals determined by the BEVENT L signal. The BEVENT L signal is obtained from the power supply via module pin BR1 at 16 2/3 ms or 20 ms intervals, depending on the line frequency source (60 Hz or 50 Hz, respectively).

### **Line Clock Status Register**

The contents of the line clock status register consist of a single read/write bit. The program communicates with the line clock via a status register at address 777 546. Program recognition of this register, along with the recognition of the boot/diagnostic registers and ROM can be enabled or disabled by the J15 jumper on the module. A second wire-wrap (J11) jumper, when installed, forces the line clock interrupt enable bit to the set condition. The line clock status register bit assignment is shown below.

**Line Clock Status Register (LKS) 777 546****Bit: 15:7****Function:** Unused**Bit: 6 (Read/write)****Name:** LINE CLOCK Interrupt Enable

**Function:** When Set, this bit allows the BEVENT line to initiate program interrupt requests. When clear, line clock interrupts are disabled. LCIE is cleared by Power-up and BINIT. LCIE is held set when the LTC ENJ L jumper is installed.

**Bit: 5:0****Function:** Unused**Bootstrap/Diagnostic ROM**

The bootstrap and diagnostic logic features three hardware registers and two ROM sockets for 2K, 4K, or 8K read-only memory. This 16-bit read-only memory contains diagnostic programs, plus a selection of bootstrap programs. These programs are user-selectable by setting eight switches on a 16-pin Dual In-Line Pack (DIP) switch pack. Programming the bootstrap and diagnostic logic consists of setting the switches for the programs desired. The bootstrap/diagnostic switch configurations and console operator responses are described in the Configuration section of this chapter. The diagnostic programs test the processor, memory, the console terminal, and the device to be bootstrapped.

The user may replace the standard ROMs with 2716-type ROMs containing programs of his choice, but the PDP-11/23-PLUS will have the functionality described above only if the standard bootstrap /diagnostic ROMs are installed in the ROM sockets.

Table 22-1 lists the error messages associated with the PDP-11/23-PLUS ROMs.

**Table 22-1 List of Error Halts**

<b>Address of Error</b>	<b>Display (Octal)</b>	<b>Cause of Error</b>
173036	01	CP1ERR, R0 contains address of error
173040	05	SLU switch selection incorrect. Error in switches
173046	05	SLU error, CSR address for selected device in error. Check CSR for selected device in floating CSR address area
173200	12	ROM Loader error. Checksum on data block
173232	02	Memory error 2. Write address into itself
173236	01	CP3ERR, R0 points to cause of error
173240	01	CP4ERR, R0 points to address of error
173262	02	Memory error 3. Byte addressing Error
173302	02	Memory error in pre-memory data test. R2 = Failing Data R3 = Expected Data R5 = Failing Address (0-776)
173316	02	Memory error. Bit 15 set in one of the parity CSR's (172100-172136). Failing memory should have the parity light on
173364	12	ROM Loader error. Checksum on address block
173376	12	ROM Loader error. Jump address is odd
173526	05	RL01/RL02 device error

Address of Error	Display (Octal)	Cause of Error
173652	05	RK05 device error
173654	01	Switch mode halt. Match was not made with switches
173660	02	Memory error in 0-2044 KW 22-bit memory test. Common error halt for six different tests  If R3=0, then error in tests 1-5 R4 determines failing test. R4 = Expected Data R5 = Failing Data

Contents of R4	Test #	Test Description
20000-27776	1	Address test, bit 11 to 0
177777	2	Data Test
000000	3	Data Test
072527	4	Odd Parity Pattern Test
125125	5	Byte Addressing Test

For tests 1-5, (R3=0) determine 22-bit failing address as follows:

R1 Bits 11-0 = Failing address bits 11-0.

R2 Bits 15-6 = Failing address bits 21-12

Errors in address uniqueness test. Test checks address bits 21-6, Test #6. If R3 is not equal to 0, then error is in this test.

R4 = Expected Data.

R5 = Failing Data

R2 = 22-bit failing physical address bits 21-6. Failing address bits 5-0 are always 0.

173664	02	Memory error in pre-memory address test for locations 000-776 R2 = Failing data R5 = Failing address and expected data
173670	01	Error in CPU test 9. JSR R3 failed
173700	01	Error CPU Test 9. JSR PC failed

Address of Error	Display (Octal)	Cause of Error
173704	05	RX01/RX02 device error
173714	04	A "NO" typed in console terminal test
173736	02	Memory error 1, Data test failed Test 0-30 Kw with MMU off if present R1 = Failing address R4 = Expected data (either 0 or 177777) R5 = Failing data
173740	01	Error CPU Test 9. "RTS" return failed
173742	03/04	Console terminal test. No done flag
173760	05	TU58 error halt

### Bootstrap and Diagnostic Registers

The bootstrap and diagnostic logic contains three hardware registers that are software-addressable. One of the registers is a dual-purpose register which functions as the configuration register when read, and the display register when written. These registers are assigned separate, individual addresses that cannot be changed or modified. The particular designations and addresses of these registers are listed in Table 22-2, below. The registers and associated logic are described in the following paragraphs.

**Table 22-2 Bootstrap and Diagnostic Register Assignments**

Register	Read/ Write	Bit Size	Address
Page Control	W	12	777520
Read/Write Maint.	R/W	16	777522
Configura- tion*	R	8	777524
Display*	W	4	777524

\* Dual-Purpose Register



The diagnostic LED display is normally used when there is no printout on the terminal after a failure. It indicates the type of error when a failure occurs in a diagnostic test or bootstrap program. The display will indicate the type of errors described in Table 22-4, below. Figure 22-4 depicts the display register.

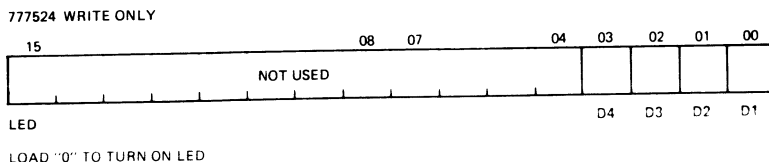


Figure 22-4 Display Register 777524

Standard DIGITAL ROMs must be installed in ROM sockets E126 (low byte) and E127 (high byte) respectively to obtain the described errors.

Table 22-4 Diagnostic LED Error Display\*

Display (Octal)	MSD		LSD		Type of Error
	Bit 3	Bit 2	Bit 1	Bit 0	
01	Off	Off	Off	On	CPU Error
02	Off	Off	On	Off	Memory
03	Off	Off	On	On	Console Terminal test display error
04	Off	On	Off	Off	Console Terminal test keyboard error
05	Off	On	Off	On	Load device status error.
06	Off	On	On	Off	Bootstrap code incorrect (NOP Instruction not in location 000000)
07	Off	On	On	On	SLU error, no response from host
10	On	Off	Off	Off	SLU received done flag set
11	On	Off	Off	On	SLU message received in DDCMP format

Display (Octal)	MSD		LSD		Type of Error
	D1 Bit 3	D3 Bit 2	D4 Bit 1	D5 Bit 0	
12	On	Off	On	Off	ROM bootstrap error
13	On	Off	On	On	Attempted access to memory Loc. #0-6 failed. This failure will normally occur when the memory test is not run and memory does not reply
17	On	On	On	On	Halt switch on or unable to run

- \* The light pattern indicates the corresponding test is in progress or failed. Some tests retry (DECnet), and others will halt the CPU (CPU, memory non-DECnet boots).

### \*Contents of R7 after halt.

#### Serial Line Units

The two full-duplex asynchronous serial line units (console serial line unit and the second serial line unit) provide the PDP-11/23-PLUS (KDF11-B) with an EIA interface which is RS-232-C and RS-423 compatible. The serial line baud rates are determined by a clock signal from an internal baud rate generator or an external clock signal via connector J1 and J2. Jumpers are provided to select either the internal clock or the external clock. If the internal clock is jumper-selected, the serial line baud rates are switch-selectable from 50 to 19.2 K baud. The console serial line and the second serial line may operate at different baud rates. However, a split baud rate is not possible. Each serial line must transmit and receive data at the same baud rate. If desired, 20 ma active or passive current loop operation at 110 baud may be obtained with DLV11-KA EIA to 20 ma converter option. The DLV11-KA contains a 110 baud rate clock signal which is supplied to pin 1 or the console serial line J1 and the second serial line J2 connectors.

The console serial line unit may be configured to halt in response to a break signal received from the console terminal. Both serial lines interrupt the processor at bus interrupt priority request level 4 (BR4).

The character format for each of the serial line units selected by wire-wrap jumpers and may consist of 7 or 8 data bits, 1 or 2 stop bits, parity or no-parity, and even and odd parity.

The console serial line unit is connected to the console terminal via connector J1. The second serial line unit is connected to the line printer, the TU58 cassette tape, or an additional terminal via connector J2.

### Serial Line Unit Registers

The program communicates with and transfers data to and from the external peripheral devices via four registers associated with each serial line. Two of the registers (RCSR and TCSR) contain control/status information for receiver and transmitter operation. The other two registers (RBUF and TBUF) contain data received from, and data to be transmitted to the peripheral device. The addresses assigned to the console and second serial line registers are listed in Table 22-5.

**Table 22-5 Serial Line Register Vectors**

Console Serial Line		Second Serial Line		
Register	Address*	Register	Address	
RCSR1	777560	RCSR2	776500**	776540***
RUF1	777562	RBUF2	776502	776542
TCSR1	777564	TCSR2	776504	776544
TBUF1	777566	TBUF2	776506	776546

\* DL1 DISJ L (J14) must be ungrounded.

\*\* DL2 DISJ L (J13) and DL2 ADRJ L (J12) must be ungrounded.

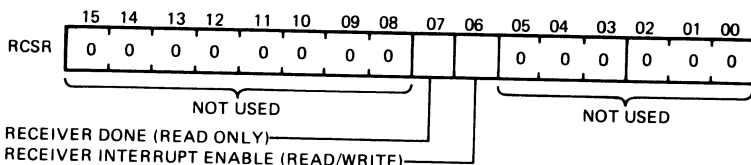
\*\*\* DL2 DISJ L (J13) must be ungrounded and DL2 ADRJ L (J12) must be grounded.

### Register Bit Assignments

The console and second serial line registers have the same bit assignments with the exception of bit 0 of the TCSR. Bit 0 is used as a transmit break bit (TX BRK) in the second serial line register (TCSR2), and it is unused in the console serial line register (TCSR 1).

The bit formats for the registers are shown in the Figures below, as are the register bit assignments.

### Receiver Status Register (RCSR 1 and RCSR 2)



**Bit:** 15:8

**Function:** Unused. Read as zeros.

**Bit:** 7 (Read-only)

**Name:** Receiver Done (RX DONE)

**Function:** Set when an entire character has been received and is ready to be read from the RBUF register. This bit is automatically cleared when RBUF is read. It is also cleared by Power-up and Bus INIT.

**Bit:** 6 (Read/Write)

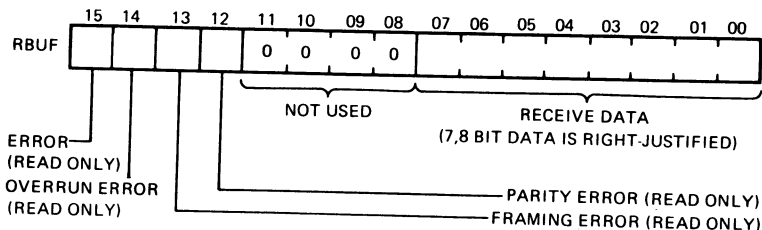
**Name:** Receiver Interrupt Enable (RX IE)

**Function:** Cleared by Power-Up and Bus INIT. If both RCVR DONE and RCVR INT ENB are set, a program interrupt is requested.

**Bit:** 5:0

**Function:** Unused. Read as zeros.

### Receiver Buffer Register (RBUF 1 and RBUF 2)



**Bit:** 15 (Read-only)

**Name:** Error (ERR)

**Function:** Set if any RBUF bit 14:12 is set. ERR is clear if all RBUF bits 14:12 are clear. This bit cannot generate a program interrupt.

**Bit:** 14 (Read-only)

**Name:** Overrun Error (OVR ERR)

**Function:** Set if a previously received character was not read before being overwritten by the present character.

**Bit:** 13 (Read-only)

**Name:** Framing Error (FRM ERR)

**Function:** Set if the present character had no valid stop bit. Also used to detect a break condition.

**Bit:** 12 (Read-only)

**Name:** Parity Error (PAR ERR)

**Function:** Set if received parity does not agree with expected parity. Always 0 if no parity is selected.

#### NOTE

Error conditions remain present until the next character is received, at which point, the error bits are updated. The error bits are cleared by Power-Up and Bus INIT.

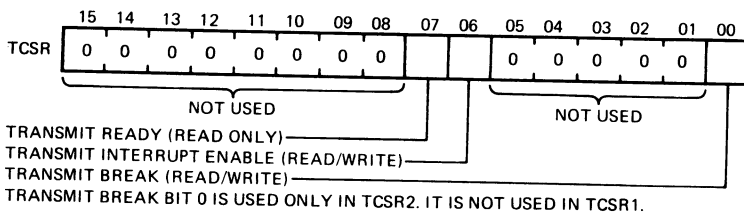
**Bit:** 11:8

**Function:** Unused. Read as zeros.

**Bit:** 7:0 (Read-only)

**Name:** Received Data Bits.

**Function:** These bits contained the last received character. If less than eight bits are selected, the character will be right-justified with the most significant bit(s) reading zero.

**Transmitter/Control Status Register (TCSR 1 and TCSR 2)**

**Bit:** 15:8

**Function:** Unused. Read as zeros.

**Bit:** 7 (Read-only)

**Name:** Transmitter Ready. (TX RDY)

**Function:** Cleared when TBUF is loaded and sets when TBUF can receive another character. XMT RDY is set by Power-up and Bus INIT.

**Bit:** 6 (Read/Write)

**Name:** Transmitter Interrupt Enable. (TX IE)

**Function:** Cleared by Power-up and Bus INIT. If both XMT RDY and XMT INT ENB are set, a program interrupt is requested.

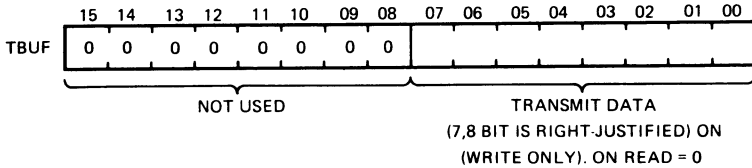
**Bit:** 5:1

**Function:** Unused. Read as zeros.

**Bit:** 0 (Read/Write)

**Name:** Break. (TX BRK)

**Function:** When set, transmits a continuous space. This bit is cleared by Power-up and System INIT. This bit is used only in TCSR2. It is unused in TCSR1.

**Transmitter Buffer Register (TBUF 1 and TBUF 2)**

**Bit:** 15:8

**Function:** Unused. Read as zeros.

**Bit:** 7:0 (Write-only)

**Function:** TBUF bits 7:0 are write-only bits used to load the transmitted character. If less than eight bits are selected, the character must be right-justified.

**CONFIGURATION****Jumper and Switch Configuration**

The PDP-11/23-PLUS contains two Dual In-Line Pack (DIP) switch units (E102 and E114) and several jumpers which allow the user to select the module features desired. The location of the switch units and jumpers is shown in Figure 22-5. The Boot/Diagnostic switch unit (E102) consists of eight switches that let the user select boot and diagnostic programs. The second switch unit (E114) selects the baud rate for the Console SLU and the second SLU. The module contains both wire-wrap jumper stakes and soldered-in jumpers. The jumpers are divided into the following functional groups.

1. Test Jumpers
2. CPU Option Jumpers
3. Device Selection Jumpers
4. Boot and Diagnostic ROM Jumpers
5. SLU Character Format Jumpers
6. Internal/External SLU Clock Jumpers
7. LSI-11 Bus Backplane Jumpers





**Manufacturing Test Jumpers**

Five jumpers are provided for manufacturing testing purposes. These jumpers must be configured as follows in Table 22-6 for normal operation.

**Table 22-6 Manufacturing Test Jumpers**

<b>JUMPER</b>		
<b>From</b>	<b>To</b>	<b>Status</b>
J6	J7	Installed
J8	J9	Installed
J20	J21	Installed
J35	J34	Installed
J33	J34	Removed
J27	J26	Installed
J25	J26	Removed

**CPU Option Jumpers**

Four wire-wrap stakes provide user-selectable features associated with the operation of the CPU. The ground stake can be connected to any combination of the other three stakes to select the available features. Two power-up mode stakes select one of three power-up modes. The halt/trap stake selects the halt/trap options.

**Power-Up Mode Selection**

The three power-up modes are available for user selection. Selection is made by installing or removing wire-wrap jumpers between jumper stakes (J17, J19) and the ground stake (J18) in various combinations. The jumper configurations for the modes are described in the Table 22-7 below.

**Table 22-7 Power-Up Mode Jumper Configurations**

<b>Mode</b>	<b>Name</b>	<b>Jumper J19</b>	<b>J18 to J17</b>
0	PC@24, PS@26	R*	R
1	Console ODT	R	I
2	Bootstrap	I	R
3	Not Implemented	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<sub>8</sub> and 26<sub>8</sub> 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 - Start at 773000**

This mode causes the processor to internally generate a bootstrap starting address of 773000 in 16-bit mode with MMU off. This address is loaded into the PC. The processor sets the PS to 340<sub>8</sub> (PS <07:05> = 7<sub>8</sub>) 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.

#### **Halt/Trap Option — J16**

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 (J16) is tested. If the jumper is removed, the processor enters console ODT mode. If the jumper is connected to J18 (ground), a trap to location 10<sub>8</sub> will occur.

#### **NOTE**

In user mode a HALT instruction execution will always result in a trap to location 10<sub>8</sub>.

**SLU Connector Pin Functions**

<b>Pin</b>	<b>J1 = Console Signal</b>	<b>J2 = Second SLU Function</b>
1	EXT CLK	Input for an external SLU clock.*
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.** Fused at 1A.

\* If the DLV11-KA option is installed between the console SLU and the console terminal, jumper J42-J43 must be removed and jumper J42-J41 installed.

\*\* This pin must be unterminated if the DLV11-KA option is not installed.

**Console SLU Switch and Jumper Configurations**

Four switches of a 16-pin DIP switch pack (E114) and four jumpers provide user-selectable features associated with the operation of the console serial line unit. A jumper is available to disable the console SLU.

**Console SLU Baud Rates**

Switches 1 through 4 of the S2 switch pack (E114) select 1 of 16 possible SLU baud rates if the internal baud rate generator is used as the clock source. If the module is configured to operate the SLU with an external clock, the positions of these switches are meaningless. The SLU transmits and receives at the selected baud rate. Split baud operation is not provided. The switch configurations to select any one of the available baud rates are listed in Table 22-10 below:

**Table 22-10 Console SLU Baud Rate Selection**

<b>Switch Position</b>				<b>Baud Rate</b>
<b>S2-4</b>	<b>S2-3</b>	<b>S2-2</b>	<b>S2-1</b>	
ON	ON	ON	ON	50
ON	ON	ON	OFF	75
ON	ON	OFF	ON	110
ON	ON	OFF	OFF	134.5
ON	OFF	ON	ON	150
ON	OFF	ON	OFF	300
ON	OFF	OFF	ON	600
ON	OFF	OFF	OFF	1200
OFF	ON	ON	ON	1800
OFF	ON	ON	OFF	2000
OFF	ON	OFF	ON	2400
OFF	ON	OFF	OFF	3600
OFF	OFF	ON	ON	4800
OFF	OFF	ON	OFF	7200
OFF	OFF	OFF	ON	9600
OFF	OFF	OFF	OFF	19,200

As stated previously, the UART can be configured to operate at a baud rate that is generated externally. The baud rate is inputted to the module from the external device through connector J1, pin 1. The jumper options are shown below.

<b>Jumper Rate</b>			<b>Selected Baud</b>	
<b>From</b>	<b>To</b>	<b>Function</b>	<b>Internal</b>	<b>External</b>
J43	J42	Connects internal baud rate generator to the console SLU UART (Normal configuration)	I	R
J41	J42	Connects external clock to the console SLU UART	R	I

### **Console SLU Character Formats**

Five wire-wrap stakes select options to establish the console SLU character format. The ground stake can be connected to any combination of the other four stakes to configure the character format options.

The following table, Table 22-11, describes how to configure the character format.

**Table 22-11 Console SLU Character Jumper Configurations**

<b>Jumper From</b>	<b>To J38**</b>	<b>Character Format Option</b>
J39	IN	7-bit characters
	OUT	8-bit characters
J37	OUT	Two stop bits
	IN	One stop bit
J36*	IN	Parity check enabled
	OUT	Parity check disabled
J40	IN	Odd parity if J36 is in
	OUT	Even parity if J36 is in

\* If 8-bit characters (J39 OUT) are selected, parity check must be disabled (J36 OUT).

\*\* J38 is the ground source for these functions. "In" means that a jumper connection is to be made to J38.

### **Break Halt Jumpers**

Two jumpers enable and disable the Break Halt feature. If this feature is enabled, the detection of a break condition by the console UART causes the processor to halt and enter the octal debugging technique (ODT) microcode. If this feature is disabled, there is no response to the break condition. Below is Table 22-12, which lists the Break Halt Jumper configurations.

**Table 22-12 Break Halt Jumper Configuration**

<b>Jumper</b>		<b>Function</b>	<b>Enabled</b>	<b>Break Feature Disabled</b>
<b>From</b>	<b>To</b>			
J5	J4	Connects ground to RQ HLT H.	R*	I*
J3	J4	Connects DL1 FE H to RQ HLT H.	I	R

\*R = removed; I = installed

### **Second SLU Switch and Jumper Configurations**

The second SLU is configured much the same as the console SLU, except that a different set of switches and jumpers are used to select the available SLU features. Also, the Halt/Break jumper is not present. Jumpers are also available to select the second SLU, and to select the

range of addresses/vectors to be used. The switch positions for the second SLU baud rates are listed in Table 22-13.

**Table 22-13 Second SLU Baud Rate Selection**

Switch Position				Baud Rate
S2-8	S2-7	S2-6	S2-5	
ON	ON	ON	ON	50
ON	ON	ON	OFF	75
ON	ON	OFF	ON	110
ON	ON	OFF	OFF	134.5
ON	OFF	ON	ON	150
ON	OFF	ON	OFF	300
ON	OFF	OFF	ON	600
ON	OFF	OFF	OFF	1200
OFF	ON	ON	ON	1800
OFF	ON	ON	OFF	2000
OFF	ON	OFF	ON	2400
OFF	ON	OFF	OFF	3600
OFF	OFF	ON	ON	4800
OFF	OFF	ON	OFF	7200
OFF	OFF	OFF	ON	9600
OFF	OFF	OFF	OFF	19,200

The second SLU may be configured to operate at an externally generated baud rate. The baud rate is inputted to the module from the external device through J2, pin 1. The jumper options are:

Jumper Rate			Selected Baud	
From	To	Function	Internal	External
J46	J45	Connects internal baud rate generator to the second SLU UART. (Normal configuration)	I	R
J44	J45	Connects external clock to the second SLU UART.	R	I

### Second SLU Character Formats

Five wire-wrap stakes select options to establish the second SLU character format. The ground stake can be connected to any combination of the other four stakes to configure the character format options.

The jumper stake functions are:

<b>Jumper From</b>	<b>To J30</b>	<b>Character Format Option</b>
J31	IN	7-bit characters
	OUT	8-bit characters
J29	OUT	Two stop bits
	IN	One stop bit
J28*	IN	Parity check enabled
	OUT	Parity check disabled
J32	IN	Odd parity if J28 is in
	OUT	Even parity if J28 is in

\* If 8-Bit characters (J31 OUT) are selected, parity check must be disabled (J28 OUT).

\*\* J30 is the ground source for these functions. "IN" means that a jumper connection is to be made to J30.

### **Boot/Diagnostic Switches and Jumpers**

A 16-pin DIP switch pack (E102) and two jumpers on the PDP-11/23-PLUS CPU module provide switch-selectable bootstrap and diagnostic programs for hard and floppy disks or the customer's own bootstrap program. The PDP-11/23-PLUS will have the functionality as described in the following sections only if 2K x 8-bit Diagnostic/Bootstrap (DIGITAL) ROMs are installed in sockets E126 and E127.

### **Bootstrap/Diagnostic Configuration Switches**

Switches S1-1 through S1-4 are used to select a diagnostic and/or a bootstrap program. Switches S1-5 through S1-8 are used in conjunction with switches S1-3 and S1-4 to select the specific bootstrap program desired.

<b>S1 Switch #</b>	<b>Configuration</b>	<b>Function</b>
1	ON	When on, execute CPU diagnostic upon power-up or restart
2	ON	When on, execute memory diagnostic upon power-up or restart

S1 Switch #	Configuration	Function
3	OFF	When on, select DECnet boot (S1-4 through S1-7 are arguments*)
4	ON	Select console test and dialogue (S1-3 OFF)  When off, select turnkey boot dispatched by S1-5 through S1-8 configuration (S1-3 OFF)

\* DECnet boot arguments are:

Boot Device**	Switch	Positions		
	S1-4	S1-5	S1-6	S1-7
DUV11	ON	X	X	X
DLV11-E	OFF	ON	X	OFF
DLV11-F	OFF	ON	X	ON

\*\*DLV11-E CSR = 775610                      DUV11 CSR = 760040 if no devices from 760010 to 760036  
DLV11-F CSR = 776500

X = Don't Care

All boots other than the above DECnet boots are controlled by the bit patterns in the switches S1-5 through S1-8 or, if the console test is selected, a mnemonic and unit number. The console test prompts with:

#### TESTING MEMORY

XXXX.KW

START?

Where XXXX is the decimal multiple of 1024 words of RAM found in the system when sized from 0 up in 1024 word increments. The first word of each 1024 word segment is read, then written back into itself. The sizing routine only sizes consecutive memory from location 0 and up.



NOTE: "Testing memory" is only printed if the memory test is selected, Turnkey mode is not selected and memory size is greater than 32 KW with memory management present.

Allowed responses are a two-character mnemonic with a one digit octal unit number (default 0) or one of two special single-character mnemonics. The response must be followed by a return, the special single-character mnemonics are:

'Y'                                      Use switch settings in S1-5 through S1-8 to determine boot device

OR 'N'                                    HALT - Enter Microcode ODT

#### "Turnkey" Bootstrap Program Selection

Switches	S1-5	S1-6	S1-7	S1-8	Program	Mnemonic
					<b>Selected</b>	
	OFF	OFF	OFF	ON	RK05 Boot	DKn n<8
	OFF	OFF	ON	OFF	RL01 or RL02 Boot	DLn n<4
	OFF	OFF	ON	ON	TU58 (SLU at 776500) Boot	DDn n<2
	OFF	ON	OFF	OFF	RX01 Boot	DXn n<2
	OFF	ON	ON	OFF	RX02 Boot	DYn n<2

#### NOTE

All bootstraps assume Unit #0, if not specified, and DIGITAL "Standard" device addresses (listed in Appendix A.)

#### Boot and Diagnostic ROM Jumpers

Two 24-pin sockets (E126 and E127) are provided for the installation of 2K x 8 ROMs or EPROMs. When EPROMs are inserted in the two ROM sockets, +5 volts must be applied to pin 21 of each socket. For all other ROMs used in this option, ROM address bit 13 (BTRA 13 H) must be applied to pin 21. Note that DIGITAL supplied devices may be either ROM or EPROM type.

Jumper		Memory Type	
From	To	ROM	EPROM
J24	J23	I*	R*
J22	J23	R	I

\* I = Installed, R = Removed

### LSI-11 Bus Backplane Jumpers

Two soldered jumpers must be installed when the PDP-11/23-PLUS is used in a backplane in which the LSI-11 Bus (or extended LSI-11 Bus) is connected to both the AB and CD rows. DIGITAL supplied backplanes of this type include: H9270, H9275 and the DDV11-B. The jumpers provide continuity for the interrupt acknowledge (BIAK) and direct memory access grant (BDMG) LSI-11 Bus signals.

Jumper	Function (when installed)
W1	Connects backplane pins CM2 and CN2 providing continuity for BIAK L
W2	Connects backplane pins CR2 and CS2 providing continuity for BDMG L

### NOTE

If the KDF11-B is installed in a LSI-11/CD Backplane (H9273-A) and the W1 and W2 jumpers are in, pin CM1 is shorted to CN1 and pin CR1 is shorted to CS1 on slot 2. Therefore, do not install peripherals in the slot immediately following the KDF11-B if they use these lines.

**Table 22-14 Factory Jumper Configuration**

Jumper	Jumper Name	Jumper State	Function
W1	BIAK	I	Provides backplane continuity for BIAK signal. Must be installed when a LSI-11/LSI-11 backplane is used
W2	BDMG	I	Provides backplane continuity for BDMG signal. Must be installed when a LSI-11/LSI-11 backplane is used

<b>Jumper</b>		<b>Jumper Name</b>	<b>Jumper State</b>	<b>Function</b>
<b>From</b>	<b>To</b>			
J22	J23	+5 V	R	When EPROMs are used, jumper J24 to J23 is removed and jumper J22 to J23 is installed
J24	J23	BTRA 13 H	I	Connects ROM address bit 13 to pin 21 of both ROM sockets (E126 and E127)
J3	J4	DL1 FE H	R	Enables BREAK HALT feature. The detection of a break condition by the console SLU causes the processor to halt and enter ODT
J5	J4	DL1 FE H	R	No halt on break
J6	J7	Master Clock	I	Enables internal master clock — do not remove
J8	J9	PHASE	I	Connects PHASE signal to F11 chip clock drivers — do not remove
J11	J10	LTC ENBJ L	R	Allows BEVENT signal to request interrupts only if bit 6 in the Line Clock Register (777546) is set
J12	J10	DL2 ADRJ L	R	Selects the following device and vector addresses for the second SLU

<b>Device Addresses</b>		<b>Vector Addresses</b>	
RCSR	776500	Receiver	300
RBUF	776502	Transmitter	304
XCSR	776504		
XBUF	776506		

J13	J10	DL2 DISJ L	R	Enables the DL2 ADRJ L jumper to determine the device and vector addresses for the second SLU
-----	-----	------------	---	---

<b>Jumper</b>		<b>Jumper Name</b>	<b>Jumper State</b>	<b>Function</b>
J14	J10	DL1 DISJ L	R	Selects the following device and vector addresses for the console SLU
		<b>Device Addresses</b>		<b>Vector Addresses</b>
		RCSR	776560	Receiver 060
		RBUF	776562	Transmitter 064
		XCSR	776564	
		XBUF	776566	
J15	J10	BDK DISJ L	R	Enables boot/diagnostic registers, boot/diagnostic ROMS and the line clk register
J16	J18	TRAP OPJ L	R	Enter console ODT if the processor is executing a HALT instruction and the processor is in Kernel mode
J17	J18	PUP CDOJ L	R	Power-up code bit 0      Bootstrap Power-up
J19	J18	PUP CDIJ L	I	Power-up code bit 1      Mode 2
J20	J21	XTL H	I	Connects baud rate oscillator to the baud rate generator. Removed for manufacturing test only
J25	J26	RCV IN	R	Console Loop-back test disabled
J27	J26	XMIT OUT	I	Connects console SLU output to the console SLU connector
J28	J30	DL2 PARJ L	R	Disable second SLU character parity check
J29	J30	DL2 ST1J L	I	Second SLU character contains one stop-bit
J31	J30	DL2 CH7J L	R	Second SLU character contains 8 bits
J32	J30	DL2 ODDJ L	R	Second SLU parity check disabled by DL2 ODDJ L

<b>Jumper</b>	<b>Jumper Name</b>	<b>Jumper State</b>	<b>Function</b>	
J33	J34	DCOKC2B H	R	Installed only during manufacturing testing
J35	J34	LINITF (1) H	I	BINIT L clears console SLU
J36	J38	DLI PARJ L	R	Disables console SLU character parity check
J37	J38	DLI ST1J L	I	Console SLU character contains one stop bit
J39	J38	DLI CH7J L	R	Console SLU character contains 8 bits
J40	J38	DLI ODDJ L	R	Console parity check disabled by DLI PARJ L
J41	J42	EXT CLK1 H	R	Disconnects EXT CLK1 input from the console SLU
J43	J42	INT CLK1 H	I	Connects baud rate clock to the console SLU
J44	J45	EXT CLK2 H	R	Disconnects EXT CLK2 input from the second SLU
J46	J45	INT CLK2 H	I	Connects baud rate clock to the second SLU

**Table 22-15 Bootstrap/Diagnostic Factory Switch Configuration**

<b>Switch S1 Number</b>	<b>(E102) Position</b>	<b>Function</b>
1	ON	Execute CPU diagnostic upon power-up or restart
2	ON	Execute Memory Diagnostic upon power-up or restart
3	OFF	DECnet boot disabled
4	ON	Console test and dialogue Enabled
5	OFF	
6	OFF	

<b>Switch S1 Number</b>	<b>(E102) Position</b>	<b>Function</b>
7	ON	RL01/RL02 Bootstrap Program selected
8	OFF	

**NOTE**

With the switch configuration shown, the KDF11-B upon power-up or restart, will execute the CPU diagnostic, the Memory diagnostic and then enter the Console test. If the operator wishes to terminate the Memory diagnostic and immediately enter the Console test, the Control/C keys must be depressed on the console terminal. If the memory test is terminated before completion, some memory locations may have wrong parity written into them.

**Table 22-16 SLU Baud Rate Factory Switch Configuration**

<b>Switch S2 Number</b>	<b>(E114) Position</b>	<b>Function</b>
1	ON	
2	OFF	Console SLU set for 9600 baud per Table 22-10
3	OFF	
4	OFF	
5	ON	
6	OFF	
7	OFF	Second SLU set for 9600 baud per Table 22-13
8	OFF	



## CHAPTER 23

### MSV11-P MEMORY

MSV11-P memory modules are MOS, random access memories (RAM), designed to be used in conjunction with the LSI-11 Bus and the Extended LSI-11 Bus.

The MSV11-P provides storage for 18-bit words (16 data bits and 2 parity bits), and contains parity control circuitry and a control and status register.

There are two MSV11-P memory modules--the MSV11-PL and the MSV11-PK. Because its capacity exceeds the capabilities of an 18-bit (LSI-11 Bus) system, the MSV11-PL can only be used in a 22-bit (Extended LSI-11 Bus) system. The MSV11-PK can also be used in a 22-bit system or an 18-bit system, if it is the only memory being used in these systems. The two MSV11-P memory modules are shown below.

<b>Model</b>	<b>Module Designation</b>	<b>Storage Capacity</b>	<b>MOS Chips</b>
MSV11-PL	M8067-LA	256K by 18 bits	64K
MSV11-PK	M8067-KA	128K by 18 bits	64K

#### FEATURES — BENEFITS

- Quad-height module — totally self-contained, compact modularity.
- Full parity functionality — detects and flags errors on a current cycle (in the event of a failure), eliminates the processing of faulty data, and prevents the execution of faulty code.
- 18- or 22-bit addressing — allows the support of up to 4 megabytes of RAM, per system.
- LED (red and green) indicator lights — to speed troubleshooting and to prevent inadvertent removal of the module while power is on.



**SPECIFICATIONS****Power**

	<b>Supply Volt- age</b>	<b>Typical Operating Power*</b>	<b>Typical Standby Power*</b>
MSV11-PK	+5 V $\pm$ 5% to- tal system power	3.45 A	3.00 A
	+5 V Battery backup**	1.75 A	1.35 A
MSV11-PL	+5 V $\pm$ 5% to- tal system power	3.60 A	3.10 A
	+5 V Battery backup**	1.85 A	1.45 A

\*These current values represent measured typical values, not maximum values.

\*\*Voltages are partitioned; however, battery backup is not supported.

**Operational****Access and Cycle Times\***

Bus Cycles	Notes		T <sub>acc</sub> (ns)+		T <sub>cyc</sub> (ns) $\pm$	
	T <sub>acc</sub> +	T <sub>cyc</sub> $\pm$	Meas Typ	Max	Meas Typ	Max
DATI	2	4	240	260	560	590
DATO(B)	2	5	90	120	610	640
DATIO(B)	3	6	660	690	1175	1210
REFRESH	-	7	-	-	640	690

\* Parity - CSR configurations, refer to notes 1, 8, and 9.

+T<sub>acc</sub> = access time

$\pm$ T<sub>cyc</sub> = cycle time

**Notes**

1. Assuming memory not busy and no arbitration.
2. SYNCH to RPLYH with minimum times (25/50 ns) from SYNCH to (DINH/DOUTH). The DATO(B) access and cycle times assume a

minimum 50 ns from SYNCH to DOUTH inside memory receivers. For actual LSI-11 Bus measurements, a constant (K-50 ns) where  $K = 200$  ns should be added to DATO(B) times, i.e.,  $T_{acc}(\text{Typ}) = 90 + (200 - 50) = 240$  ns.

3. SYNCH to RPLYH (DATO(B)) with min time (25 ns) from SYNCH to DINH and minimum (350 ns) from RPLYH (DATI) asserted to DOUT asserted.
4. SYNCH to TIM250H negated.
5. SYNCH to TIM250H negated with minimum time (50 ns) from SYNCH to DOUTH.
6. SYNCH to TIM250H (DATO(B)) with minimum times (25 ns) from SYNCH to DINH and minimum (350 ns) from RPLYH (DATI) asserted to DOUT asserted.
7. Ref REQ L to TIM250H negated.
8. REFRESH arbitration adds (90 ns) Typ and (110 ns) Maximum to access.
9. REFRESH conflict adds (640 ns) typ and (690 ns) maximum to access and cycle times.

## Environmental

### Temperature

#### Storage Temperature Range

-40°C to +66°C (-40°F to +151°F)

Do not operate a module that has been stored outside the operating temperature range. Bring the module to an environment within the operating range and allow at least five minutes for the module to stabilize.

#### Operating Temperature Range

+5°C to +60°C (40°F to 140°F)

Lower the maximum operating temperature by 1°C for each 1000 feet of altitude above 8000 feet.

### Relative Humidity

#### Storage

10% to 90%, non-condensing

#### Operating

10% to 90%, non-condensing

### **Operating Airflow**

Adequate airflow must be provided to limit the inlet to outlet temperature rise across the module to +5°C when the inlet temperature is 60°C (140°F). For operation below +55°C airflow must be provided to limit the inlet to outlet temperature rise across the module to 10°C maximum.

### **Altitude**

#### **Storage**

The module will not be mechanically or electrically damaged at altitudes up to 50,000 feet (90 MM mercury).

#### **Operating**

Up to 50,000 feet (90 MM mercury).

Note: Lower the maximum operating temperature by 1°C for each 1000 feet of altitude above 8000 feet.

## **CONFIGURATION**

The jumpers on the MSV11-P memory module are divided into these five groups:

- Starting Address Jumpers
- CSR Address Jumpers
- Power Jumpers
- Bus Grant Continuity Jumpers
- Miscellaneous Jumpers

### **Module Starting Address (MSA)**

Each MSV11-P memory module installed in a system is jumpered for its own starting address by the use of wire-wrapped pins. The memory module starting address can be determined by how much memory the system contains. To do this, convert the byte value (how much memory the system has), to a word value. This word value is the module starting address.

Module starting address jumpers consist of the following two groups:

1. First Address of the Range (FAR) - Selects the first 256K range address that the starting address falls in. (Table 23-1, Part 1).
2. Partial Starting Address (PSA) - Selects which 8K boundary within a specific multiple of 256K words the starting address falls in. (Table 23-1, Part 2).

After you have determined your module starting address (MSA), you should determine the FAR and PSA values.

1. Find the FAR value - This is accomplished by referring to Table 23-1, Part 1, and locating the address range the MSA falls in. The FAR value is the first address of the selected address range. Associated with the FAR value is a specific configuration of jumper pins X, W, and V that use jumper pin Y, a ground pin.
2. Find the PSA value - This is done by inserting the MSA and FAR values into the equation  $PSA = MSA - FAR$ . First, perform the necessary subtraction operation. Then, on Table 23-1, Part 2, locate the proper PSA value. Associated with the PSA is a specific configuration of jumper pins P, N, M, L, and T, all of which use jumper pin R, a ground pin.

**Table 23-1 Starting Address Configurations (Part 1)**

<b>First Address Ranges (FAR)</b>		<b>Jumpers to Ground (Pin Y)</b>		
<b>Starting Address Range (FAR)</b>		<b>Pin X</b>	<b>Pin W</b>	<b>Pin V</b>
<b>Decimal (K)</b>	<b>Octal</b>	<b>A21</b>	<b>A20</b>	<b>A19</b>
000-248	00000000-01740000	out	out	out
256-504	02000000-03740000	out	out	in
512-760	04000000-05740000	out	in	out
768-1016	06000000-07740000	out	in	in
1024-1272	10000000-11740000	in	out	out
1280-1528	12000000-13740000	in	out	in
1526-1784	14000000-15740000	in	in	out
1742-2040	16000000-17740000	in	in	in



**Table 23-1 Starting Address Configurations (Part 2)**

Partial Starting Address (PSA)		Jumpers to Ground (Pin R)				
Decimal (K)	Octal	Pin P A18	Pin N A17	Pin M A16	Pin L A15	Pin T A14
0	00000000	out	out	out	out	out
8	00040000	out	out	out	out	in
16	00100000	out	out	out	in	out
24	00140000	out	out	out	in	in
32	00200000	out	out	in	out	out
40	00240000	out	out	in	out	in
48	00300000	out	out	in	in	out
56	00340000	out	out	in	in	in
64	00400000	out	in	out	out	out
72	00440000	out	in	out	out	in
80	00500000	out	in	out	in	out
88	00540000	out	in	out	in	in
96	00600000	out	in	in	out	out
104	00640000	out	in	in	out	in
112	00700000	out	in	in	in	out
120	00740000	out	in	in	in	in
128	01000000	in	out	out	out	out
136	01040000	in	out	out	out	in
144	01100000	in	out	out	in	out
156	01140000	in	out	out	in	in
160	01200000	in	out	in	out	out
168	01240000	in	out	in	out	in
176	01300000	in	out	in	in	out
184	01340000	in	out	in	in	in
192	01400000	in	in	out	out	out
200	01440000	in	in	out	out	in
208	01500000	in	in	out	in	out
216	01540000	in	in	out	in	in
224	01600000	in	in	in	out	out
232	01640000	in	in	in	out	in
240	01700000	in	in	in	in	out
248	01740000	in	in	in	in	in

**Control and Status Register (CSR) Jumpers**

Each MSV11-P memory module contains a control and status register. The bus master can read or write the CSR via the Extended LSI-11 Bus. The CSR is a 16-bit register with an address that starts in the top 4K of system address space.

The CSR is assigned to one of the 16 addresses shown in Table 23-2. CSR addresses are assigned as follows.

1. Determine how many memory modules in your system have CSR registers.
2. List the memory modules' sequential position from the CPU.
3. The memory modules closest to the CPU have the lower module starting address (MSA).
4. The memory module with the lowest MSA is assigned to the lowest CSR address and jumpered according to Table 23-2.
5. The next sequential CSR memory module is assigned the next highest CSR address.

Each memory module has four CSR jumper pins (A, B, C, and D) which can be daisy chained to pin E, the ground pin. The jumpers allow logic to detect a specific CSR address that has been assigned to a CSR memory module.

For example, assume the system has two memory modules with CSR registers. You are installing the third CSR memory. Refer to Table 23-2 and find the column labeled module number three. The CSR jumper pin configuration is pin B wire-wrapped to pin E. The memory module's CSR address is 17772104 for large systems or 772104 for small systems.

**Table 23-2 CSR Address Selection**

Module Number	Large System	Small System	Jumper to Ground (Pin E)			
	LSI-11 Bus Address	LSI-11 Bus Address	D	C	B	A
1	17772100	772100	out	out	out	out
2	17772102	772102	out	out	out	in
3	17772104	772104	out	out	in	out
4	17772106	772106	out	out	in	in
5	17772110	772110	out	in	out	out
6	17772112	772112	out	in	out	in
7	17772114	772114	out	in	in	out
8	17772116	772116	out	in	in	in
9	17772120	772120	in	out	out	out
10	17772122	772122	in	out	out	in
11	17772124	772124	in	out	in	out
12	17772126	772126	in	out	in	in
13	17772130	772130	in	in	out	out
14	17772132	772132	in	in	out	in
15	17772134	772134	in	in	in	out
16	17772136	772136	in	in	in	in

### **Power Jumpers**

The power jumpers for the M8067-LA and M8067-KA are configured as follows:

- W3, W10, W11 — Always Out
- W4, W5, W9, W13, W15 — In for Non Battery Backup
- W4, W5, W12, W14 — In for Battery Backup

For an illustration of these power jumpers, please refer to Figure 23-1, the MSV11-P Memory Board, depicted above.

#### **NOTE**

DIGITAL does not support battery backup.

### **Bus Grant Continuity Jumpers**

W1 and W2 are installed only in backplanes that have the LSI-11 Bus (or Extended LSI-11 Bus) in both the A and B rows and C and D rows. DIGITAL-supported backplanes that use this scheme are the H9270 and the H9275. For use in the H9273 and H9276 backplanes, W1 and W2 are removed.

### **Miscellaneous Jumpers**

The miscellaneous jumpers are always configured as follows, and cannot be altered by the user:

Always In: 3-9, 4-10, 6-7, 13-15, 14-16, 43-44

Always Out: 8-7, F-H, 22-23, 44-45

### **System Size Jumpers**

A wire-wrap post is provided to configure the memory for the addressing size of the system. Post 2 is connected to post Y (ground) when the MSV11-P is used in an 18-bit (LSI-11 Bus) system. In 22-bit systems (Extended LSI-11 Bus), this jumper is removed.

### **DESCRIPTION**

The MSV11-P memory module consists of a single, quad-height module (M8067) that contains the Extended LSI-11 Bus interface, timing and control logic, refresh circuitry, and a MOS storage array. The module also contains circuitry to generate and check parity, and a control and status register.

All MSV11-P memory modules have the logic functions shown in Figure 23-2. The functions shown in Figure 23-2 are discussed in detail in the following paragraphs.

### **Board Selection Decode Logic**

Board Selection Decode Logic consists of two PROMs that monitor



BDAL 21-14 and compare the address received with the starting address jumpers. The PROMs are programmed to enable the logic to generate memory select (MSEL) and row enables (NA 16/18 and NA 15/17). Table 23-3 illustrates the output of the PROMs programmed for module M8067-LA with a starting address of zero. There are two different types of PROMs.

1. M8067-LA PROMs programmed for 256K address per module
2. M8067-KA PROMs programmed for 128K addresses per module

**Table 23-3 PROM Output for M8067-LA (Starting Address Zero)**

Octal Address	NA 18	NA 17	Row Enable
01777776	1	1	3
01400000			
01377776	1	0	2
Gener- 01000000			
ates 00777776			
MSEL L			
	0	1	1
00400000			
00377776			
	0	0	0
00000000			

The address range of the two PROMs begins with the starting address (jumper-selectable). The top limit of the memory module is then determined by what type of PROMs are being used (e.g., 256K from the starting address).

### **MOS Memory Address Logic**

Jumper consideration must be made for loading the row and column latches with the MOS RAM address. The logic row also has a row latch used for refresh addresses.

The MSV11-P memories perform three basic operations:

- CSR read/write transfers
- Memory read/write transfers
- Refresh cycles

When CSR transfers take place, the row select signals (RAS 0-3) and column select signals (CAS 0-3) are inhibited. Please refer to Figure 23-3.

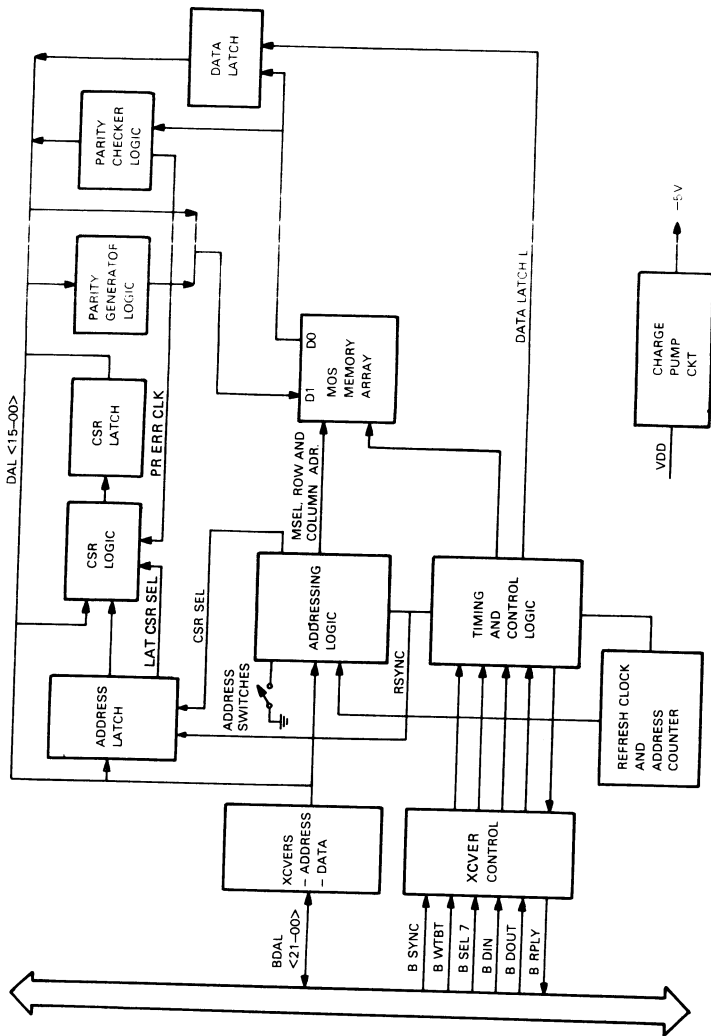


Figure 23-2 MSV11-P Memory Logic Functions

When read/write cycles take place, the PROM sends the row enable signals (RAS 0-3) to the MOS memory array in order to select the proper row. All columns (CAS 0-3) are always selected (Figure 23-3). First the row address, then the column address are loaded into internal registers in the MOS RAM chips. One 18-bit location in memory is now selected.

When refresh cycles take place, the column select logic is inhibited. All rows are selected (RAS 0-3). The refresh row address is loaded into an internal register in the MOS RAM chips (Figure 23-3), and that address is refreshed.

### **CSR Address Logic**

Memory receives a CSR address and compares (XOR) DAL 1-DAL 4 with the CSR jumpers. If there is a match, CSR selected (CSR) is generated. CSR generates MSEL, which enables a read/write request to start the memory timing. RAS and CAS are inhibited; therefore, no memory addressing occurs.

### **CSR Write**

The CSR write operation is shown in Figure 23-4. CSR address and signals BBS7 and BWTBT are received by the memory. If the CSR address matches the CSR jumpers, CSR SEL and MSEL are generated. When BSYNC is received, the address and CSR SEL are latched, and the WT REQ flip-flop is set. During the data cycle the memory receives the data to be stored in the CSR register. Then, BDOUT is received and the write request starts memory timing. The signal LAT CSR SEL selects the received data to be passed through the multiplexed inputs to the CSR register. The CSR clock logic generates the CSR clock pulse which, in turn, loads the the CSR register.

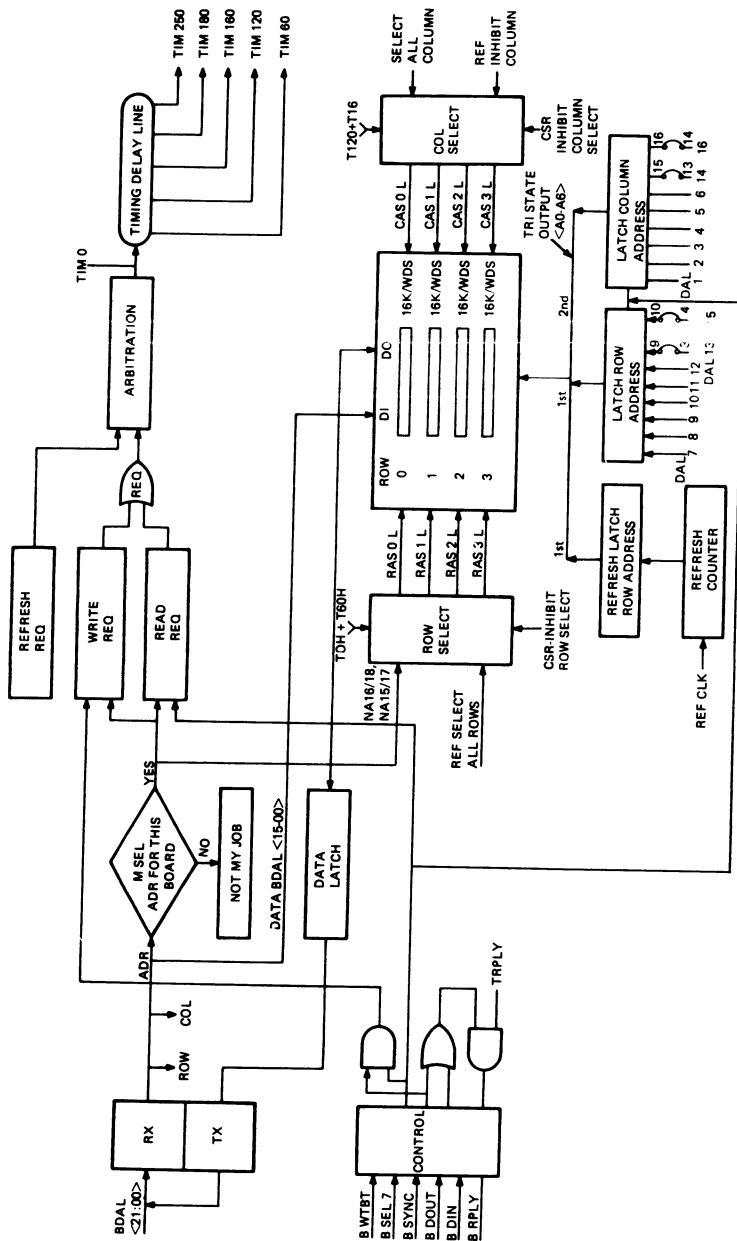


Figure 23-3 Overview of Memory Logic



### CSR Read

CSR address, BBS7 and BWTBT negated are received by the memory. If the CSR address matches the CSR jumpers, CSR SEL and MSEL are generated. When BSYNC is received, the address and CSR SEL are latched, and the RD REQ flip-flop is set, which starts the memory timing. As soon as the TRPLY is generated, the contents of the CSR are latched. Memory receives BDIN L, which enables the memory to transmit the latched CSR data onto the Extended LSI-11 Bus. Please refer to Figure 23-5, the CSR Read diagram.

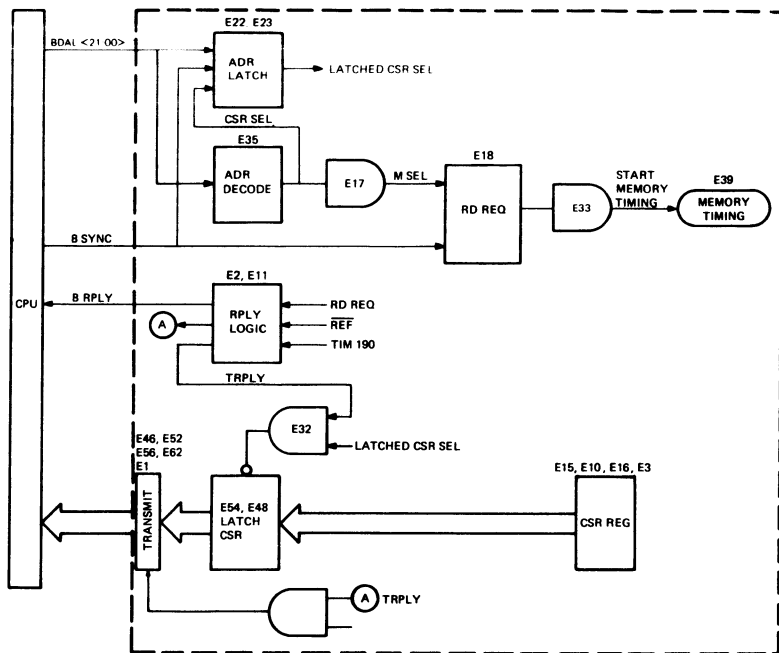


Figure 23-5 CSR Read

### Memory Array

The MSV11-P memories use early writes. Early writes are achieved by a write going low prior to CAS going active. Data in is strobed into the MOS chips by CAS going active.

The following MOS chips are used in the MSV11-P memory modules:

- M8067-LA (64K chips)- A fully populated module that consists of four rows of MOS RAM chips (512 KB)

- M8067-KA (64K chips)-A half populated module that consists of two rows of MOS RAM chips (256 KB)

The MOS RAM chips used in the M8067-LA and M8067-KA memory modules are dynamic random access memory circuits organized as a 65,536 by 1 bit (Figure 23-6).

### Timing and Control Logic

The memory responds to the asynchronous read/write commands or the synchronous refresh cycle that takes place every 14.5 microseconds. All requests go to the memory timing lockout gate. When timing lockout is negated, the request goes to arbitration and the winner gains control of the memory timing (Figure 23-7).

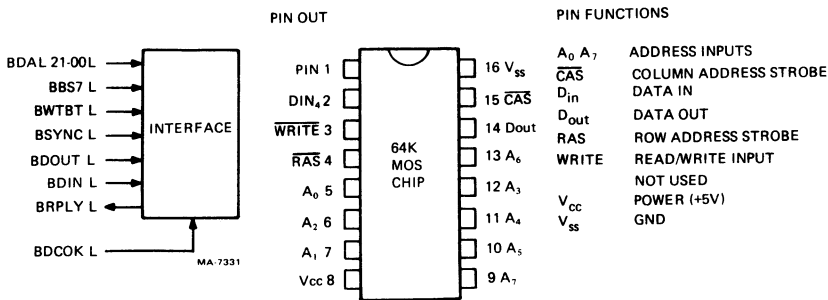


Figure 23-6 64K MOS RAM chip

### Parity Control Circuitry

The parity control circuitry in the MSV11-P generates parity bits based on the data being written into memory during a DATO or DATBO bus cycle. One parity bit is assigned to each data byte, and is stored with the data in the MOS storage array. When data is retrieved from memory during DATI or DATIO bus cycles, the parity of the data is determined. If parity is good, the data is assumed correct. If the parity bits do not correspond, the data is assumed unreliable and memory will initiate the following action:

- The red LED is lit. This provides visual indication of a parity error and sets CSR bit 15.
- If bit 0 in the CSR is set, the memory asserts BDAL 16 and 17. This warns the processor that a parity error has occurred.
- Part of the address of the faulty data is recorded in the CSR

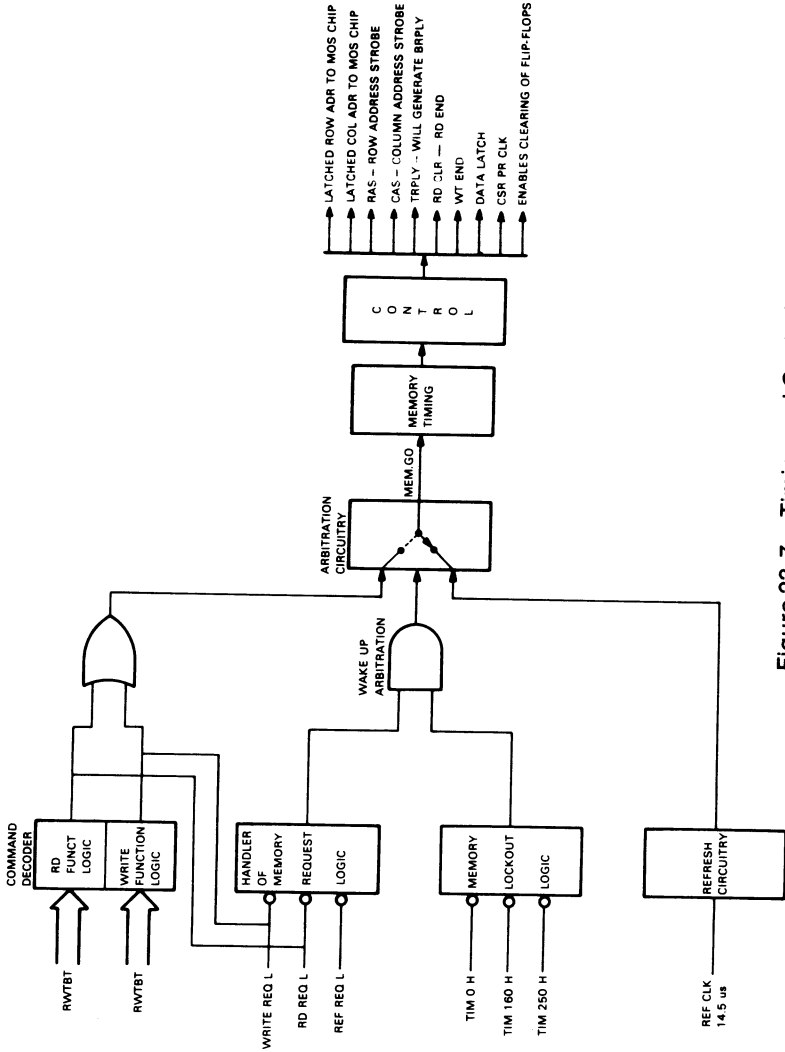


Figure 23-7 Timing and Control



### Parity Logic

The parity logic performs these two functions:

1. Parity generation- when the bus master is doing a DATO(B)
2. Parity checking- when the bus master is doing a DATI

### Parity Generation

The data received from the bus master on the LSI-11 Bus (DAL00-15) goes to the parity generators (Figure 23-8).

The low-byte parity generator generates odd parity (PDI 16 H). The high-byte parity generator generates even parity (PDI 17 H).

CSRO2 enables the diagnostic to force wrong parity. This enables the diagnostic to check out the parity logic.

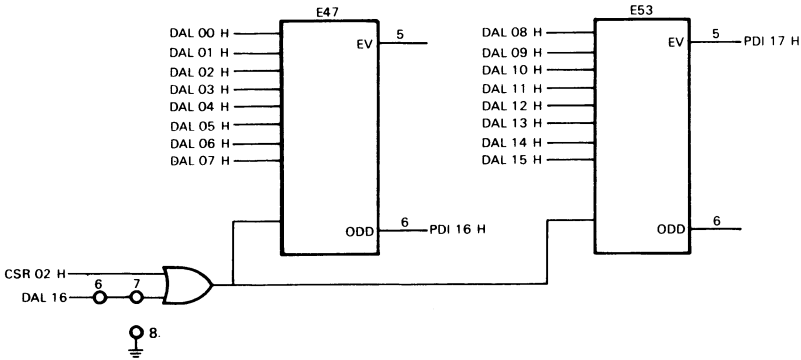


Figure 23-8 Parity Generators

### Parity Checkers

The MSV11-P memory modules detect parity errors, report parity errors, and save the address of the parity error in the CSR register. Parity detection logic always expects byte 0 to have an even number of one bits and byte 1 to have an odd number of one bits. If this does not happen, T PAR ERR L is generated (Figure 23-9).

Parity reporting is done if the program has set CSR bit 0, the parity error enable bit, T PAR ERR L, and RDIN negated. This allows BDAL 16 and BDAL 17 to be sent out on the LSI-11 Bus to flag a parity error.

Parity address is saved in the following way:

- After the data is read from memory, it goes to the parity checkers and a holding register.
- When the data is latched, the signal CSR CLK is generated if there is a parity error (Figure 23-9).
- CSR CLK latches the address in the CSR register.
- CSR cycles or refresh cycles inhibit the parity logic.

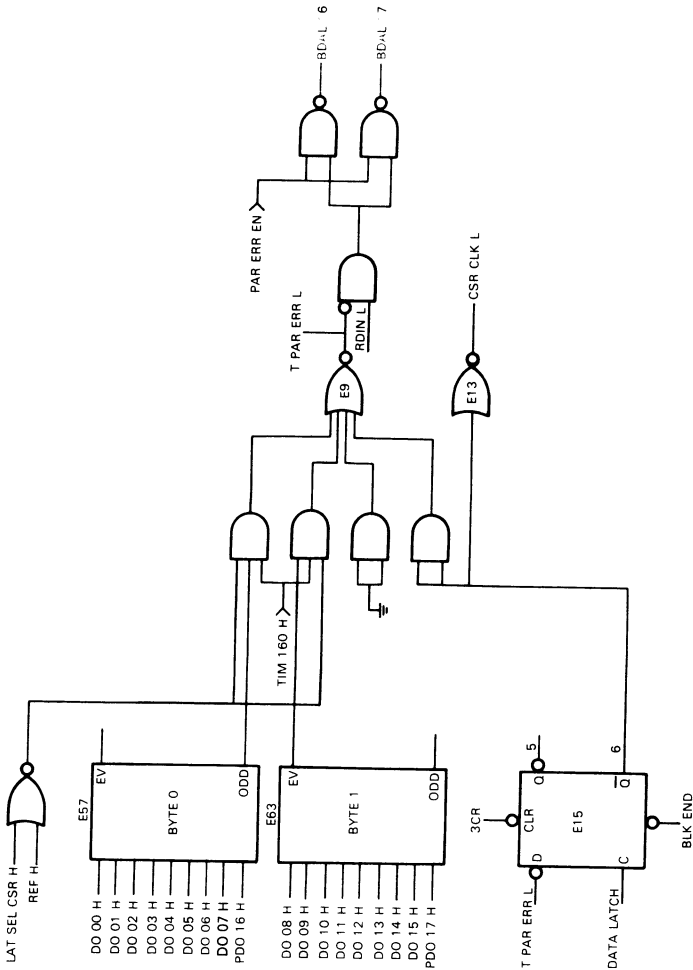


Figure 23-9 Parity Checkers

## Refresh

The MSV11-P memory modules use MOS RAM chips which are re-freshed every 2 ms. The logic refreshes a row at a time; 128 rows are refreshed in a 2 ms period.

Figure 23-10 shows that a refresh timer generates a pulse every 14.5 microseconds. This pulse, called REF CLK, does the following:

- Generates a refresh request
- Increments the refresh address CTR

The address counter produces row address bits A1 through A8 for 64K chips.

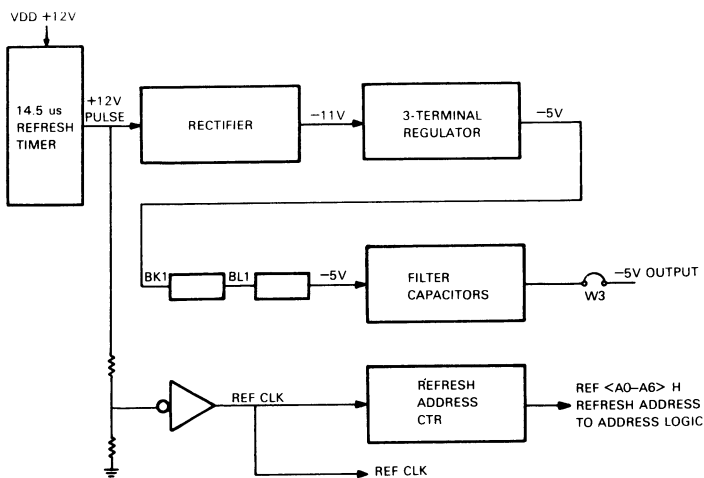


Figure 23-10 Refresh Logic

## Charge Pump Circuit

The purpose of the charge pump circuit is to generate a filtered regulated  $-5\text{ V}$  for possible future use. The power jumpers, described above, effectively disconnect this circuit.

## Control and Status Register

The control and status register in the MSV11-P memory module contains bits that are used to store the parity error address bits. By setting a bit in the CSR, you can force wrong parity. This is a useful diagnostic tool for checking out the parity logic. The CSR has its own address in the top 4K of memory. Bus masters can read or write to the CSR.

The CSR allows programs control of certain parity functions, and contains diagnostic information if a parity error has occurred. The CSR is assigned an address and can be accessed by a bus master via the LSI-11 Bus. Some CSR bits are cleared by assertion of BUS INIT L. This signal is asserted for a short time by the processor after system power has come up, or in response to a reset instruction. The CSR bit assignments are shown in Figure 23-11, and are described as follows:

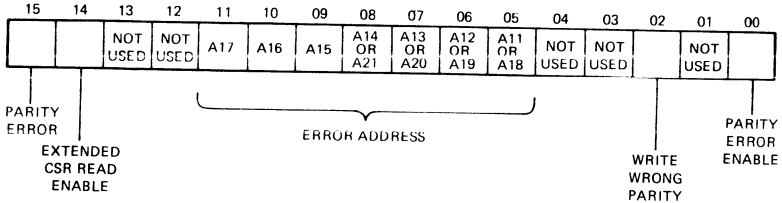


Figure 23-11 CSR Bit Allocation

Bits 1, 3, 4,  
12, 13

**Not used.** These bits are always read as logical zeros. Writing to these bits has no effect on the CSR.

Bit 0

**Parity Error Enable.** If set, and a parity error occurs on a DATI or DATIO(B) cycle to memory, then BDAL 16 (L) and BDAL 17 (L) are asserted on the bus at the same time as the data. This bit is read/write and is reset to zero on power-up or BUS INIT.

Bit 2

**Write Wrong Parity.** If this bit is set, it is equivalent to 1, a DATO or DATOB cycle to memory will occur, and wrong parity data will be written into the parity MOS RAMs. This bit may be used to check the parity error logic as well as failed address information in the CSR. The following diagnostic is applicable:

1. With bit 2 set, write entire memory with any pattern.
2. Read first location in memory. If bit 0 of the CSR is set, then a parity error should be detected on the LSI Bus and the failed address at location (0) is stored in the CSR.

3. Read the CSR and obtain the failed address. If CSR bit 14 equals 0, then bits A11-A17 are loaded into CSR bits 5-11. If CSR bit 14 equals 1, then bits A18-21 have been loaded into CSR bits 5-8. Bit 2 is a read/write bit reset to zero on power-up or BUS INIT.

Bits 05-11

**Error Address Bits.** If a parity error occurs on a DATI or DATIO(B) cycle, then A11-A17 are stored in CSR bits 5-11 and bits A18-A21 are latched. The 18-bit address machines require only one read of the CSR to obtain the failed address. The CSR bit set to 0 allows the logic to pass A11-A17 to the LSI-11 Bus. A 22-bit machine requires two reads. The first read (with CSR bit-14 set to 0) sends contents to CSR bits 5-11. Then the software must set CSR bit-14 equal to 1 to enable A18-A21 to be read from CSR bits 5-8.

The parity error address locates a parity error to a 1K segment of memory. These are read/write bits and are not reset to zero via power-up or BUS INIT. If a second parity error is found, the new failed address will be stored in the CSR.

Bit 14

**Extended CSR Read Enable.** For a functional description of what this bit does, refer to the error address description.

1. Bit 14=0; always for 128K word machine
2. Bit 14=0; first read on 2048K word machine
3. Bit 14= 1; second read on 2048K word machine

Bit 15

**Parity Error.** When set, this bit indicates that a parity error has occurred. The bit lights a red LED on the module, providing visual indication of a parity error. Bit 15 is a read/write bit. It is reset to zero via power-up or by BUS INIT. It will remain set unless rewritten or initialized.

**section IV**  
**appendices**

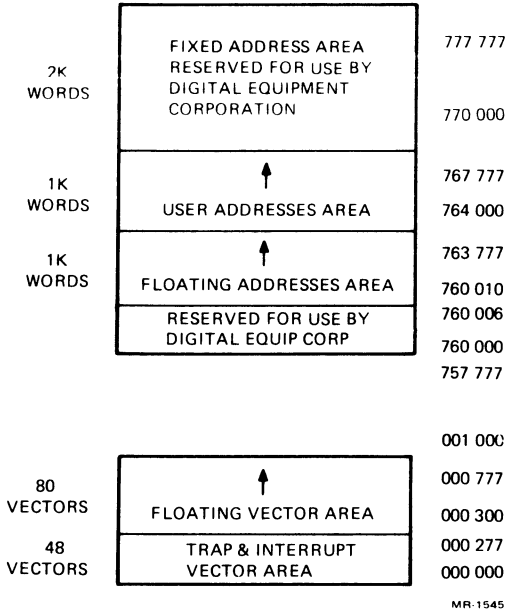




# APPENDIX A

## ASSIGNMENT OF ADDRESSES AND VECTORS

### ADDRESS MAP



### FLOATING VECTORS

The conventions for the assignments of floating vectors for modules on the LSI-11 bus will adhere to those established for UNIBUS devices. UNIBUS devices are used to explain the priority ranking for floating vectors and are included in the subsequent table of trap and interrupt vectors as a guide to the user.

The floating vector convention used for communications and for devices that interface with the PDP-11 series of products assigns vectors sequentially starting at 300 and proceeding upward to 777. (Some LSI-11 bus modules, such as the DLV11 and DRV11, have an upper vector limit of 377). The following table shows the sequence for assigning vectors to modules. It can be seen that the first vector address, 300, is



*Appendix A—Assignment of Addresses and Vectors*

assigned to the first DLV11 in the system. If another DLV11 is used, it would then be assigned vector address 310, etc. When the vector addresses have been assigned to all the DLV11s (up to a maximum of 32), addresses are then assigned consecutively to each unit of the next highest ranked device (DRV11 or DLV11-E, etc.), then to the other devices according to their rank.

**Ranking for Floating Vectors  
(Start at 300 and proceed upward.)**

<b>Rank</b>	<b>UNIBUS</b>	<b>LSI-11 Bus</b>
1	DC11	
2	KL11, DL11-A, -B	DLV11, -F, -J
3	DP11	
4	DM11-A	
5	DN11	
6	DM11-BB	
7	DR11-A	DRV11-B
8	DR11-C	DRV11
9	PA611 Reader	
10	PA611 Punch	
11	DT11	
12	DX11	
13	DL11-C, -D, -E	DLV11-E
14	DJ11	
15	DH11	
16	GT40	
17	LPS11	
18	DQ11	
19	KW11-W	KWV11
20	DU11	DUV11


**INTERRUPT AND TRAP VECTORS**

<b>Vector</b>	<b>UNIBUS</b>	<b>LSI-11 Bus</b>
000	DEC reserved	DEC reserved
004	CPU errors	Bus time-out and illegal instructions (e.g., JMP R0) (odd address and stack overflow traps not
.		
.		
.		

**INTERRUPT AND TRAP VECTORS (Cont)**

<b>Vector</b>	<b>UNIBUS</b>	<b>LSI-11 Bus</b>
.		implemented on LSI-11)
010	Illegal and reserved instructions	Illegal and reserved instructions
014	BPT, breakpoint trap	BPT instruction and T bit
020	IOT, input/output trap	IOT instruction
024	Power-fail	Power-fail
030	EMT, emulator trap	EMT instruction
034	TRAP instruction	TRAP instruction
040	System software	
044	System software	
050	System software	
054	System software	
060	Console terminal, keyboard/reader	Console terminal, input
064	Console terminal, printer/punch	Console terminal, output
070	PC11, paper tape reader	
074	PC11, paper tape punch	
100	KW11-L, line clock	External event line interrupt
104	KW11-P, programmable clock	
110		
114	Memory system errors	
120	XY plotter	
124	DR11-B DMA interface; (DA11-B)	DRV11-B
130	AD01, A/D subsystem	
134	AFC11, analog subsystem	
140	AA11, display	
144	AA11, light pen	
150		
154	Reserved for Digital	
160	RL11	RLV11
164	Reserved for Digital	
170	User reserved	
174	User reserved	
200	LP11/LS11, line printer; LA180	LAV11, LPV11
204	RS04/RF11, fixed head disk	

**INTERRUPT AND TRAP VECTORS (Cont)**

Vector	UNIBUS	LSI-11 Bus
210	RC11, disk	
214	TC11, DECTape	
220	RK11, disk	RKV11
224	TU16/TM11/TS03, magnetic tape	
230	CD11-CM11-CR11, card reader	
234	UDC11, digital control subsystem	
240	PIRQ, program interrupt request (11/45)	
244	Floating-point error	FIS (optional)
250	Memory management	Memory Management
254	RP04/RP11 disk pack	
260	DIP11/TA11/TU78	
264	RX11, floppy disk	RXV11, RXV21
270	User reserved	} User reserved
274	User reserved	
300	(Start of floating vectors)	
374		} ADV11-A
400		
404		
410		
414		
420		} IBV11-A
424		
430		
434		
440		} KVV11-A
444		
450	} User reserved	
777		
		(End of floating vectors)

### FLOATING ADDRESSES

The conventions for the assignment of floating addresses for modules on the LSI-11 bus are the same as UNIBUS devices. UNIBUS devices are used to explain the ranking sequence.

The floating address convention used for communications and for other devices that interface with the PDP-11 series of products assigns addresses sequentially starting at 760 010 (or 160 010) and proceeds upward to 763 776 (or 163 776). For compatibility with UNIBUS convention, addresses are expressed as consisting of 18 bits (7XX XXX) rather than 16 bits (1XX XXX).

Floating addresses are assigned in the following sequence:

Rank	UNIBUS Device	LSI-11 Device
1	DJ11	DUV11
2	DH11	
3	DQ11	
4	DU11	
5	DUP11	
6	LK11A	
7	DMC11	DZV11
8	DZ11	
9	KMC11	RLV11 (extras)
10	RL11 (extras)	

### DEVICE ADDRESSES

Address	UNIBUS	LSI-11 Bus
777 776	Processor status word (PS)	
777 774	Stack limit	
777 772	Program interrupt request (PIRQ)	
777 770	DIGITAL reserved	
.		
.		
777 720		

**DEVICE ADDRESSES (Cont.)**

Address	UNIBUS	LSI-11 Bus
777 716	} CPU Registers R7 (PC) R6 (SP) R5 R4 R3 R2 R1 R0 } General Registers	
777 710		
777 707		
777 706		
777 705		
777 704		
777 703		
777 702		
777 701		
777 700		
777 676	} Memory management	
777 600		
777 576	(SR2)	Memory mgt status register
777 574	(SR1)	
777 572	(SR0)	
777 570	Console switch and display register	
777 566	(XBUF)	
777 564	(XCSR)	} Console Terminal
777 562	(RBUF)	
777 560	(RCSR)	
777 556		
777 554	} PC11/PR11	
777 552		
777 550		
777 546	KW11-L, DL11-W	(LTC) KP11 BDV11
777 544	} XY11	
777 530		
777 526		Unassigned

Appendix A—Assignment of Addresses and Vectors

DEVICE ADDRESSES (Cont.)

Address	UNIBUS	LSI-11 Bus
777 524	Unassigned	BDV11
777 522		
777 520		
777 516	LA180, LP11 LS11, LV11	LAV11, LPV11
777 514		
777 512		
777 510		
777 506	TA11	
.		
777 500		
777 476	RF11	
.		
777 460		
777 456	RC11	
.		
777 440		
777 436	DT11, bus switch	#8
777 434		#7
777 432		#6
777 430		#5
777 426		#4
777 424		#3
777 422		#2
777 420		#1
777 416	RK11	RKV11
.		
777 400		

**DEVICE ADDRESSES (Cont.)**

Address	UNIBUS	LSI-11 Bus
777 376	DC14-D	
.		
777 360		
777 356	TC11	
.		
777 340		
777 336	KE11-A, EAE #2	
.		
777 320		
777 316	KE11-A, EAE #1	arithmetic shift logical shift normalize step count/status register multiply multiplier quotient accumulator divide
777 314		
777 312		
777 310		
777 306		
777 304		
777 302		
777 300		
777 276	DIGITAL reserved	
.		
777 200		
777 176	RX11	} RXV11
777 174		
777 172		
777 170	-RX11-	} RXV11, RXV21
777 166	CR11, CM11, CD11	
777 164		
777 162		
777 160		

**DEVICE ADDRESSES (Cont.)**

Address	UNIBUS	LSI-11 Bus
777 156	DIGITAL reserved	<div style="display: flex; align-items: center; justify-content: center;"> <div style="border-left: 1px dashed black; padding-left: 10px; text-align: center;"> <p>This area reserved for 16 serial line units <i>without</i> modem control capability</p> </div> <div style="margin: 0 10px;"> <p>↑</p> <p>↓</p> </div> </div>
...		
777 000		
776 776		
...	AD01	
776 770		
776 766		
...	AA11 #1	
776 750		
776 746		
...	Unassigned	
776 740		
776 736		
...	RP11	
776 700		
776 676	DL11-A, -B #4-#16	
...		
776 530		
776 526		
776 524	DL11-A, -B, #3	
776 522		
776 520		
776 516	DL11-A, -B #2	
776 514		
776 512		
776 510		



DEVICE ADDRESSES (Cont.)

Address	UNIBUS	LSI-11 Bus
776 506	} DL11-A,-B, #1	↕
776 504		
776 502		
776 500		
776 476	} AA11	↕
...		
776 400	} DX11	↕
776 376		
...		
776 200		
776 176	} #6-#31	↕
...		
775 660		
775 656		
775 654	} #5	↕
775 652		
775 650	} #4	↕
775 646		
775 644		
775 642		
775 640	} DL11-C,-D,-E	↕
775 636		
775 634	} #3	↕
775 632		
775 630	} #2	↕
775 626		
776 624		
775 622		
775 620		

This area reserved for 31 serial line units with modem control capability

Appendix A—Assignment of Addresses and Vectors

DEVICE ADDRESSES (Cont.)

Address	UNIBUS	LSI-11 Bus
775 616	} #1	} ↑ ↓
775 614		
775 612		
775 610		
-----		
775 606	} DIGITAL reserved	
775 604		
775 602		
775 600		
776 576	} #4	
.		
.	} DS11	
.		
775 400	} #1	
775 376		
.	} #16	
.		
.	} DN11	
.		
775 200	} #1	
775 176		
.	} #16	
.		
.	} DM11	
.		
775 000	} #1	
774 776		
.	} #1	
.		
.	} DP11	
.		
774 410	} DP11	
774 406		
774 404	} RL01	} RLV11
774 402		
774 400	} #32	
774 376		
.	} #32	
.		
.	} DC11	
.		
774 000	} #1	



Appendix A—Assignment of Addresses and Vectors

DEVICE ADDRESSES (Cont.)

Address	UNIBUS	LSI-11 Bus
772 776	PA611 typeset punch	
.		
772 700		
772 676	PA611 typeset reader	
.		
772 600		
772 576	AFC11	
772 574		
772 572		
772 570		
772 566		
.	DIGITAL reserved	
.		
772 560		
772 556	DIGITAL reserved	
.		
.		
772 550		
772 546	KW11-P	
772 544		
772 542		
772 540		
772 536	TM11	
772 534		
772 532		
772 530		
772 526		
772 524		
772 522		
772 520		

DEVICE ADDRESSES (Cont.)

Address	UNIBUS	LSI-11 Bus
772 516	Memory mgt status register (SR3)	
772 514		
.	OST	
.		
772 500		
772 456	DR11-B, #3	
.		
772 450		
772 446	TJU16	
.		
772 440		
772 436	DR11-B, #2	DRV11-B, #3
772 434		
772 432		
772 430	DIGITAL reserved	DRV11-B, #2
772 426		
772 424		
772 422	DR11-B,-C, #1	DRV11-B, #1
772 420		
772 416		
772 414	KW11-W	
772 412		
772 410		
772 406	Memory management	
.		
772 400		
772 376		
.		
772 200		

*Appendix A—Assignment of Addresses and Vectors*

**DEVICE ADDRESSES (Cont.)**

Address	UNIBUS	LSI-11 Bus
772 176	} FP11	
.		
.		
772 160		
772 156	} Unassigned	
.		
.		
772 140		
772 136	} Memory parity	
.		
.		
772 110		
772 106	} Unassigned	
.		
.		
772 102		
772 100	MS11-K, -LP, MM11-LP	
772 076	} RL11	
.		
.		
772 070		
772 066	} RJS04	
.		
.		
772 040		
772 036	} DIGITAL reserved	
.		
.		
772 020		

Appendix A—Assignment of Addresses and Vectors

DEVICE ADDRESSES (Cont.)

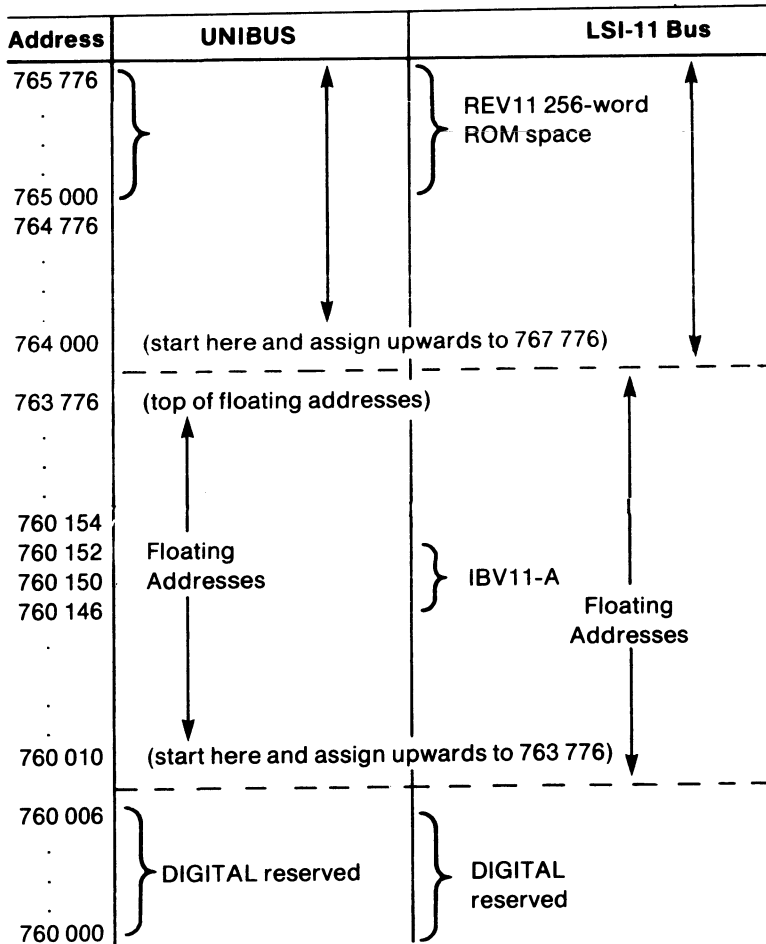
Address	UNIBUS	LSI-11 Bus
772 016	GT40, VT48	
...		
772 000		
771 776	UDC functional I/O modules	
...		
771 000		
770 776	#8	
...	KG11	
770 700		
770 676	#1	
...	#16	
770 500	DM11-BB	
770 476		
...	#1	
770 460	ADF11	
770 456		
...	Unassigned	
770 450		
770 446		
770 444		
770 442	LPS11	} AAV11-A
770 440		
770 436		

Appendix A—Assignment of Addresses and Vectors

DEVICE ADDRESSES (Cont.)		
Address	UNIBUS	LSI-11 Bus
...	LPS11	} KVV11-A
770 424		
770 422		
770 420		
770 416	AR11, LPS11	} ADV11-A
...		
770 404		
770 402		
770 400	DIGITAL reserved	
770 376		
...		
770 000		
767 776	} DR11-C, #1	} DRV11, #1
767 774		
767 772	} DR11-C, #2	} DRV11, #2
767 770		
767 766	} DR11-C, #3	} DRV11, #3
767 764		
767 762		
767 760		
767 756		
767 754		
767 752		
767 750		
767 746		
...	} User Reserved Area	} User Reserved Area
776 000		



**DEVICE ADDRESSES (Cont.)**



## APPENDIX B

# LSI-11/23 INSTRUCTION TIMING

**Appendix B consists of four sections:**

- Instruction Execution Time—Standard PDP-11 & EIS
- Floating Point Instruction Timing
- DMA Latency
- Interrupt Latency

### LSI-11/23 INSTRUCTION EXECUTION TIME— STANDARD PDP-11 & EIS

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.

$$\text{INSTR TIME} = \text{Basic Time} + \text{SRC Time} + \text{DST Time}$$

$$(\text{BASIC TIME} = \text{Fetch Time} + \text{Decode Time} + \text{Execute Time})$$

Some of the instructions require only some of these times.

The tables that follow list typical instruction execution times for standard memories. These times assume typical gate delays, the maximum MSV11 access time, refresh overhead, 40 ns. bus propagation delays, and a 300 ns. microcycle. Times can vary  $\pm 20\%$ . All timing is in microseconds unless otherwise noted.

If memory management is enabled, add .16  $\mu\text{s}$ . for each memory reference. To arrive at incremental value to add to the instruction times given in the tables, use the following equation:

$$\text{Increment} = .16 (\text{reads and writes}) + .32 (\text{read/modify/write})$$

Select numbers from the memory cycle column in the table below to insert in this equation.

**BASIC TIMES**  
Memory Cycles

Instructions	Micro-Cycles	R	W	RMW	MSV11-C	MSV11-D	MSV11-E
MOV, CMP, BIT, ADD, SUB, BIC, BIS, XOR, SXT, CLR, TST, SWAB, COM, INC, DEC, NEG, ADC, SBC, ROR, ROL, ASR, ASL, MFPS	4	1			1.85 us.	1.72 us.	1.76 us.
MTPS	14	1			4.85 us.	4.72 us.	4.76 us.
MFPI (D)	12	1			4.25 us.	4.12 us.	4.16 us.
MTPI (D)	6	2			3.10 us.	2.85 us.	2.93 us.
MUL	78-80	1			24.65 us.	24.52 us.	24.56 us.
DIV	132-167	1			50.75 us.	50.62 us.	50.66 us.
ASH	14-101	1			30.95 us.	30.83 us.	30.86 us.
ASHC	21-155	1			47.15 us.	47.02 us.	47.06 us.
All Branch Instructions	4	1			1.85 us.	1.72 us.	1.76 us.
SOB (Branch)	7	1			2.75 us.	2.62 us.	2.66 us.
(No Branch)	6	1			2.45 us.	2.32 us.	2.36 us.
RTS	7	2			3.40 us.	3.15 us.	3.23 us.
MARK	12	2			4.90 us.	4.65 us.	4.73 us.
RTI, RTT	12	3			5.55 us.	5.17 us.	5.29 us.
Set or Clear C, V, N, Z	7	1			2.75 us.	2.62 us.	2.66 us.

Instructions	Micro- Cycles	R	W	RMW	MSV11-C	MSV11-D	MSV11-D
HALT	21–25	3	2		3.05 us. 124.70 us.	2.92 us. 124.57 us.	2.96 us. 124.61 us.
WAIT	8	1			9.11 us.	7.95 us.	8.07 us.
RESET	8–430	1			10.01 us.	8.85 us.	8.97 us.
BMT, TRAP	17	3	2		2.15 us.	2.02 us.	2.06 us.
IOT, BPT	20	3	2		2.45 us.	2.32 us.	2.36 us.
JMP	5	1			3.10 us.	2.85 us.	2.93 us.
(mode 1)		2			2.45 us.	2.32 us.	2.36 us.
(mode 2)		2			3.40 us.	3.15 us.	3.23 us.
(mode 3)		2			3.40 us.	3.15 us.	3.23 us.
(mode 4)		3			4.65 us.	4.27 us.	4.39 us.
(mode 5)		1	1		4.38 us.	3.86 us.	3.90 us.
(mode 6)		1	1		4.68 us.	4.16 us.	4.20 us.
(mode 7)		2	1		5.33 us.	4.69 us.	4.77 us.
JSR	10	1	1		4.68 us.	4.16 us.	4.20 us.
	10	2	1		5.63 us.	4.99 us.	5.07 us.
	10	1	1		5.63 us.	4.99 us.	5.07 us.
	11	2	1		6.88 us.	6.11 us.	6.23 us.
	11	2	1				
	11	2	1				
	11	2	1				
	13	3	1				

Instructions	Micro-Cycles	SOURCE TIMES			MSV11-D	MSV11-E
		R	W	RMW		
Source Addressing	0				0	0
(mode 0)	2	1			1.12 us.	1.16 us.
(mode 1)	2	1			1.12 us.	1.16 us.
(mode 2)	4	2			2.25 us.	2.33 us.
(mode 3)	3	1			1.42 us.	1.46 us.
(mode 4)	5	2			2.55 us.	2.63 us.
(mode 5)	5	2			2.80 us.	2.63 us.
(mode 6)	7	3			4.05 us.	3.79 us.
(mode 7)						

DESTINATION ADDRESSING  
Memory Cycles

Instructions	Micro- Cycles	R	W	RMW	DESTINATION ADDRESSING		
					MSV11-C	MSV11-D	MSV11-E
(mode 0)	0				0	0	0
MOV, CLR, (mode 1)	4		1		2.23 us.	1.84 us.	1.84 us.
SXT, MFPS, (mode 2)	4		1		2.23 us.	1.84 us.	1.84 us.
MTPI (D) (mode 3)	5	1	1		3.18 us.	2.66 us.	2.70 us.
(mode 4)	4		1		2.23 us.	1.84 us.	1.84 us.
(mode 5)	6	1	1		3.48 us.	2.96 us.	3.00 us.
(mode 6)	6	1	1		3.48 us.	2.96 us.	3.00 us.
(mode 7)	8	2	1		4.73 us.	4.09 us.	4.17 us.
CMP, BIT, (mode 0)	0				0	0	0
(mode 1)	3		1		1.55 us.	1.42 us.	1.46 us.
(mode 2)	3		1		1.55 us.	1.42 us.	1.46 us.
(mode 3)	4	2	2		2.50 us.	2.25 us.	2.33 us.
(mode 4)	3	1	1		1.55 us.	1.42 us.	1.46 us.
(mode 5)	5	2	2		2.80 us.	2.55 us.	2.63 us.
(mode 6)	5	2	2		2.80 us.	2.55 us.	2.63 us.
(mode 7)	7	3	3		4.05 us.	3.67 us.	3.79 us.
MTPS, MFPI (D), (mode 0)	0				0	0	0
(mode 1)	-1	1			0.35 us.	0.22 us.	0.26 us.
MUL, DIV, ASH,							

Instructions	Micro- Cycles	R	W	RMW	MSV11-C	MSV11-D	MSV11-E
ASHC	(mode 2)	1			0.35 us.	0.22 us.	0.26 us.
	(mode 3)	2			1.30 us.	1.05 us.	1.13 us.
	(mode 4)	1			0.35 us.	0.22 us.	0.26 us.
	(mode 5)	2			1.60 us.	1.35 us.	1.43 us.
	(mode 6)	2			1.60 us.	1.35 us.	1.43 us.
	(mode 7)	3			2.85 us.	2.47 us.	2.59 us.
	(mode 0)	0			0	0	0
BIC, BIS,							
ADD, SUB,	5		1		3.18 us.	2.66 us.	2.70 us.
SWAB, COM,	5		1		3.18 us.	2.66 us.	2.70 us.
INC, DEC,	6	1	1		4.13 us.	3.49 us.	3.57 us.
NEG, ADV,	5		1		3.18 us.	2.66 us.	2.70 us.
SBC, ROR,	7	1	1		4.43 us.	3.79 us.	3.87 us.
ROL, ASR,	7	1	1		4.43 us.	3.79 us.	3.87 us.
ASL, XOR	9	2	1		5.68 us.	4.91 us.	5.03 us.

**FLOATING POINT INSTRUCTION TIMING**

The execution time of a KEF11-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 to the above, the execution time of many instructions, such as ADDF, are dependent on the data.

Table 1 provides the basic instruction times for addressing mode 0 with a microcycle time of 300 ns. Tables 2 through 5 show the additional time required, using the MSV11-E, for instructions with other than mode 0. Refer to the notes for the execution time variations for the data dependent instructions.

**Table 1**

Instruction	Micro-	Mode 0	Notes	Modes 1 thru 7
	cycles	Time (us.)		
LDF	28	9.15	1,2,19	Use Table 2
LDD	36	11.55	1,2,23	
LDCFD	40	12.75	1,4	
LDCDF	55	17.25	1,5	
CMPE	65	20.25	14,15	
CMPD	71	22.05	14,15	
DIVF	301	91.05	1,29,41,43,44	
DIVD	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	



Table 1 (CONTINUED)

	<b>Micro-</b>	<b>Mode0</b>		
<b>Instruction</b>	<b>cycles</b>	<b>Time (us.)</b>	<b>Notes</b>	<b>Modes 1 thru 7</b>
STF	18	6.15		Use Table 3
STD	26	8.55		
STDCDF	65	20.25	1,38	
STCFD	48	15.15	1,4	
CLRF	36	11.55		
CLRD	40	12.75		
ABSF	43	13.65	37	Use Both Tables 2 and 3
ABSD	51	16.05	37	
NEGF	42	13.35	1,37	
NEGD	50	15.75	1,37	
LDFPS	11	4.05		Use Table 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	
STFPS	16	5.55		Use Table 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		

Table 2

Addressing Mode	Microcycles*		Read/Write Memory Cycles		Time (us.)	
	Single Precision	Double Precision	Single	Double	Single Precision	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 number (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-zero
3. if the floating point number is a negative and zero with FIUV flag clear

Table 3

Addressing Mode	Microcycles*		Read/Write Memory Cycles		Time (us.)	
	Single Precision	Double Precision	Single	Double	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

Table 4

Addressing Mode	Microcycles*		Read/Write Memory Cycles		Time (us.)	
	Short Integer	Long Integer	Short	Long	Short Integer	Long 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

Table 5

Addressing Mode	Microcycles*		Read/Write Memory Cycles		Time (us.)	
	Short Integer	Long Integer	Short	Long	Short Integer	Long 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-zero.
3. Add 900 ns. if SRC > 177 or SRC < -177.
4. Add 900 ns. if floating point number = 0.
5. Add 3.3 us. if overflow on rounding.
6. Add 300 ns. if integer is negative.

7. Add 1.5 us. if absolute value of integer  $< 65,536$ .
8. Add 1.2 us.  $n$  times where  $n = 220 - \text{expn}$ .
9. Add 1.2 us.  $n$  times where  $n = 240 - \text{expn}$  and if absolute value of integer  $\geq 65,536$ .
10. Add 600 ns. if  $\text{expn} < 20$ .
11. Add 2.1 us.  $n$  times where  $n$  is the smaller of the two absolute values:  $(210 - \text{expn})$  or  $(230 - \text{expn})$ .
12. Add 600 ns. if integer is negative.
13. Add 900 ns. if integer is negative.
14. Add 1.2 us. if floating point numbers are equal.
15. Add 2.1 us. if numbers are unequal but the signs are the same.
16. Add 600 ns. if  $\text{FPACC} > \text{FPSRC}$ .
17. Add 2.4 us. if  $\text{FPSRC} > \text{FPACC}$ .
18. Add 600 ns. if adding opposite signs or subtracting like signs.
19. Add 2.4 us. if trapped on undefined variable.
20. Add 900 ns. and 1.2 us.  $n$  times where  $n = \text{expn difference}$ .
21. Add 3.6 us. if  $\text{FPSRC} > \text{FPACC}$ .
22. Add 1.2 us. if adding opposite signs or subtracting like signs.
23. Add 1.2 us. if trapped on undefined variable.
24. Add 900 ns. and 1.8 us.  $n$  times where  $n = \text{expn difference}$ .
25. Add 1.2 us.  $n$  times where  $n = \text{shifts to normalize}$
26. Add 1.8 us.  $n$  times where  $n = \text{shifts to normalize}$
27. Add 3.3 ns. if underflow.
28. Add 600 ns. if overflow.
29. Add 600 ns. if need to normalize after multiply or divide.
30. Add 1.2 us. if need to normalize after multiply or divide.
31. Add 600 ns. for every "1" bit in multiplier (FPSRC).
32. Add 1.2 us. for every "1" bit in multiplier (FPSRC).
33. Add 900 ns. times  $n$  where  $n = \text{expn mod } 16$  (calc integer and fraction).
34. Add 300, 600, or 900 ns. if  $\text{expn} = 21-40, 41-60$  or  $> 100$ , or 61-100 respectively.
35. Add 1.8 us. if the fractional part = 0.
36. Add 1.2 us. if the fractional part = 0.
37. Add 4.5 us. if trapped on any of the FP11 interrupts.
38. Add 5.4 us. if trapped on overflow.
39. Add 24.3 us. if trapped on conversion error.
40. Add 24.9 us. if trapped on conversion error.

41. Add 1.2 us. if rounding.
42. Add 1.8 us. if rounding.
43. Add 8.1 us. if trapped on overflow.
44. Add 9.0 us. if trapped on underflow.

### DMA LATENCY

DMA (Direct Memory Access) latency, which is the time from request (BDMRL) to bus mastership for the first DMA device, is:

#### DMA LATENCY

	MSV11-C
4.15 us., worst case	
	MSV11-D
3.49 us., worst case	
	MSV11-E
3.53 us., worst case	

Worst Case Time = Longest DATIO Cycle + Refresh Time

A 300 ns. microcycle time is assumed.

### INTERRUPT LATENCY

Interrupt latency is the sum of the time from request (BIRQL) to acknowledgement (BIAKL) and the time from acknowledge to fetch of the first service routine instruction.

		<b>BIRQL TO BIAKL</b>	<b>BIAKL TO FETCH</b>	<b>TOTAL WORST CASE</b>
Standard PDP-11 Instruction Set	MSV11-C	12.11 us.	9.41 us.	21.52 us.
	MSV11-D	10.79 us.	8.18 us.	18.97 us.
	MSV11-E	11.07 us.	8.26 us.	19.33 us.
EIS	MSV11-C	56.28 us.	9.41 us.	65.69 us.
	MSV11-D	55.65 us.	8.18 us.	63.83 us.
	MSV11-E	55.81 us.	8.26 us.	64.07 us.
FP11 (KEF Option)	MSV11-C	N/A	N/A	N/A
	MSV11-D	N/A	N/A	N/A
	MSV11-E	47.72 us.	8.26 us.	55.98 us.

For all floating point instructions except ADD, SUB, MUL, DIV and MOD, the interrupt latency is the length of the instruction. Unlike the KD11 processors, the EIS instructions are not interruptable. In the floating point arithmetic instructions, interrupts may be serviced during their execution. 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 us. and the time must be added to the interrupt latency times where execution of an instruction is aborted.



## APPENDIX C

# LSI-11,11 /2 INSTRUCTION TIMING

### 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.

$$\text{INSTR TIME} = \text{Basic Time} + \text{SRC Time} + \text{DST Time}$$

$$(\text{BASIC Time} = \text{Fetch Time} + \text{Decode Time} + \text{Execute Time})$$

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  $\pm 20$  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 $\mu\text{s}$	1.05 $\mu\text{s}$	2.10 $\mu\text{s}$	1.75 $\mu\text{s}$
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  $\mu\text{s}$  to SRC time and 0.70  $\mu\text{s}$  to DST time.

### BASIC TIME

DOPS (Double Operand)	DM0	DM1-7
MOV	3.50 $\mu\text{s}$	2.45 $\mu\text{s}$
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

DM0 = Destination Mode 0

DM1-7 = Destination Modes 1 through 7



*Appendix C—LSI-11, 11/2 Instruction Timing*

SOPS (Single Operand)	DM0	DM1-7
CLR	3.85 $\mu$ s	4.20 $\mu$ s
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  $\mu$ s to instr. time if Bit 7 of effective OPR = 1

JMP/JSR MODE	DST TIME
1	0.70 $\mu$ s
2	1.40
3	1.75
4	1.40
5	2.45
6	2.45
7	4.20

INSTRUCTION	BASIC TIMES
JMP	3.50 $\mu$ s
JSR (PC = LINK)	5.25
JSR (PC $\neq$ 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 * +

## Appendix C—LSI-11, 11/2 Instruction Timing

INSTRUCTION	BASIC TIMES
TRAP,EMT	16.80 * $\mu$ s
IOT,RPT	18.55 *
WAIT	6.30
HALT	5.60
RESET	5.95 + 10.0 $\mu$ s for INIT + 90.0 $\mu$ s
MAINT INST. (00021R)	20.30
RSRVD INST. (00022N)	5.95 (TO GET TO $\mu$ ADDRESS 3000)

\* If NEW PS HAS BIT 4 or BIT 7 SET ADD 0.35  $\mu$ s FOR EACH  
 + IF NEW PS HAS BIT 4 (T BIT) SET ADD 2.10  $\mu$ s

### EXTENDED ARITHMETIC (KEV11) INSTRUCTION TIMES

#### EIS Instruction Times

MODE	SRC TIME
0	0.35 $\mu$ s
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 $\mu$ s	If both numbers less than 256 in absolute value 16 bit multiply
	64.0 $\mu$ s Worst Case	
DIV	78.0 $\mu$ s 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 $\mu$ s
FSUB	42.4

EXPONENT DIFFERENCE	ALIGN BINARY POINTS
0– 7	2.45 $\mu$ s 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 $\mu$ s 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 ( $\mu$ s)	
FMUL	74.2 to 80.9 $\mu$ s if either argument has only 7 bits of precision, i.e., the second word of the 32 bit argument is 0. 121.1 $\mu$ s worst case (i.e., arguments have more than 7 bits of precision).
FDIV	151 $\mu$ s typical 232 $\mu$ s 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  $\mu$ s, 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 $\mu$ s max
Time from acknowledgement (BIAK L) to fetch of first service routine instruction	+ 16.5 $\mu$ s max
Total time from request to first service routine instruction	134.5 $\mu$ s 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 $\mu$ s max
Time from acknowledgement (BIAK L) to fetch of first service routine instruction	+ 16.5 $\mu$ s max
Total time from request to first service routine instruction	35.05 $\mu$ s 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 $\mu$ s max
Time from acknowledgement (BIAK L) to fetch of first service routine instruction	+ 16.5 $\mu$ s max
Total time from request to first service routine instruction	<hr/> 44.1 $\mu$ s 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  $\mu$ s for the interrupt service routine.



## **APPENDIX D**

# **PDP-11 FAMILY 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.

Since the PDP-11/23-PLUS instruction set is the same as the LSI-11/23 instruction set, PDP-11/23 PLUS instruction differences are listed under LSI-11/23 in the following pages of this appendix.

FALCON SBC-11/21 instruction differences are included in this appendix.

**Appendix D—PDP-11 Family Differences**

Activity	LSI-	PDP-11				FALCON SBC-11/21
		11	04	05/10	15/20	
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
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
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
OPR PC, X(R); OPR PC, @X(R); OPR PC, @A; OPR PC, A: location A will contain the PC of OPR +4.	X			X	X	X
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

## Appendix D—PDP-11 Family Differences

Activity	LSI-				PDP-11			FALCON SBC-11/21
	11	11/23	04	05/10	15/20	34	35/40	
JMP (R)+ or JSR reg. (R)+: contents of R are incremented by 2, then used as the new PC address.				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	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				
Register addresses (177700—177717) are valid program addresses when used by CPU.				X		X	X	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	X



Appendix D—PDP-11 Family Differences

FALCON  
SBC-11/21

Activity	LSI-	11/23	04	05/10	15/20	34	35/40	45	FALCON SBC-11/21
SOB, MARK, RTT, SXT instructions.	X	X	X			X	X	X	X
ASH, ASHC, DIV, MUL.	X	X				X	X	X	X
XOR instruction.	X	X				X	X	X	X
The external option KE11-A provides MUL, DIV and SHIFT operation in the same data format.				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-	11	11/23	04	05/10	15/20	34	35/40	45	FALCON SBC-11/21
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	X		X		X	
Power fail acts the same as 11/45 (22 milliseconds with about 300 nanoseconds minimum). Power fail during RESET fetch is fatal with no power down sequence.				X	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					
If RTT sets the T bit, the T bit trap occurs after the instructing following RTT.		X	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	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	X	X

**Appendix D—PDP-11 Family Differences**

**FALCON  
SBC-11/21**

Activity	LSI-	PDP-11				FALCON SBC-11/21			
		11	04	05/10	15/20				
If an interrupt occurs during an instruction and the T bit is set, the interrupt is acknowledged before T bit trap.	11	11/23	04	05/10	15/20	34	35/40	45	X
T bit trap will sequence out of WAIT instruction		X	X	X	X	X	X		X
T bit trap will not sequence out of WAIT instruction. 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	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				
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

Appendix D—PDP-11 Family Differences

Activity	LSI-11	11/23	04	05/10	PDP-11	34	35/40	45	FALCON SBC-11/21
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	15/20	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	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								X
Four interrupt levels exist.		X	X	X	X	X			X
Stack overflow not implemented.	X						X	X	
Stack overflow below 400 implemented.			X	X	X	X			
Red and yellow zone stack overflow implemented.		X	X	X	X	X			X

Activity	LSI-	PDP-11					FALCON SBC-11/21					
		11	11/23	04	05/10	15/20		34	35/40	45		
Odd address trap not implemented.	X	X						X				
Odd address trap implemented.			X						X			
FMUL and FDIV instructions implicitly 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.												
Due to their execution time, FIS instructions can abort because of a device interrupt.	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.												
MOV instruction does just a DATO bus sequence for the last memory cycle.	X											X

Appendix D—PDP-11 Family Differences

Activity	LSI-11	PDP-11				FALCON SBC-11/21			
		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	
If PC contains non-existent memory address and a bus error occurs, PC will 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			X	
Same as above but register is unchanged.		X							
If register contains an odd value in mode 2 and a bus error occurs, register will be incremented.	X	X						X	
If register contains an odd value in mode 2 and a bus error occurs, register will be unchanged.			X	X	X			X	
Condition codes restored to original values after FIS interrupt abort (EIS doesn't abort on 35/40).			X	X	X			X	

Appendix D—PDP-11 Family Differences

Activity	LSI-	PDP-11					FALCON SBC-11/21			
		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				
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
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
Op codes 075040 through 075777 trap to 10 as reserved op codes.		X		X	X	X				X

**Appendix D—PDP-11 Family Differences**

Activity	LSI-	PDP-11				FALCON SBC-11/21
		11/23	04	05/10	15/20	
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.	11 X					45 X
Op codes 170000 through 177777 trap to 10 as reserved instructions.			X	X		
Op codes 170000 through 177777 are implemented as floating point instructions.	X					34 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					35/40 X
CLR, SXT, MFPS, MTP1 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		
MEM.MGT maintenance mode SR0 bit 8 is implemented.					X	X
MEM.MGT maintenance mode SR0 bit 8 is not implemented.	X					X



FALCON  
SBC-11/21

Activity	LSI-	PDP-11					FALCON SBC-11/21
		11	11/23	04	05/10	15/20	
PS <15:12>, user mode, user stack pointer, and MTPX and MFPX instructions exist even when MEM.MGT is not configured.		X					45
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
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-	11/23	04	05/10	15/20	34	35/40	45	FALCON SBC-11/21
MIPS in user mode will access PS <7:0> regardless of whether PS address 177776 is mapped.	11	X							
A HALT instruction in user mode traps to 4.									
A HALT instruction in user mode traps to 10.		X				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	
CLR(B) and SXT do a write (DATO) bus sequence for last bus cycle									X
Resident ODT microcode									X
Event line interrupt on level 6									X
The no-BSACK 18 μ sec timeout implemented. If time-out occurs, BDMGO is aborted									X

## PRIORITY OF TRAPS AND INTERRUPTS

<b>LSI-11</b>	<b>LSI-11/23</b>	<b>PDP-11/04</b>	<b>PDP-11/05,10</b>
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 Interrupts	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		

<b>PDP-11/34</b>	<p>ODDAD Trap MMU Trap BUSERR Trap PARERR Trap Trap Inst. Trace Trap STOVF Trap PFAIL Trap Device Interrupts Console Halt Wait Loop</p>	<b>PDP-11/35,40</b>	<p>PARERR Trap MMU Trap BUSERR Trap STOVF Trap (Red Zone) Trap Inst. Trace Trap STOVF Trap (Yellow Zone) PFAIL Trap Console Halt Device Interrupts Wait Loop</p>	<b>PDP-11/45</b>	<p>Console Halt ODDAD Trap STOVF Trap (Red Zone) MMU Trap BUSERR Trap PARERR Trap STOVF Trap (Yellow Zone) PFAIL Trap PIRQ Device Interrupts Wait Loop Trace Loop</p>	<p>BUSERR = Timeout Error MMU = Memory Management Trap PARERR = Parity Error ODDAD = Odd Address Error STOVF = Stack Overflow PFAIL = Power Fail Inst. = Instructions</p>
------------------	---	---------------------	--	------------------	---	---

**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 re-

sponse 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**

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.

When an address location is open, another location can be opened without explicitly closing the first location (e.g., 1000/123456 2000/054321).

“↑” will open the previous location.

“@” will open a location using indirect addressing.

“←” will open a location using relative addressing.

“M” will print the contents of an internal CPU register.

Rubout (ASCII 177) will delete the last character typed in.

“L” is the boot loader command which will load the absolute loader.

**LSI-11/23**

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.

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 its contents.

“↑” is illegal and micro ODT prints ?(CR)(LF)@.

“@” is illegal and micro ODT prints ?(CR)(LF)@.

“?” is illegal and micro ODT prints ?(CR)(LF)@.

“M” is illegal and micro ODT prints ?(CR)(LF)@.

Rubout is illegal and micro ODT prints ?(CR)(LF)@.

“L” is illegal and micro ODT prints ?(CR)(LF)@.



**LSI-11, LSI-11/2**

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.

Up to a 16-bit address and 16-bit data may be entered. Leading zeroes are assumed.

Incrementing (LF), the address 177776 results in the address 000000.

Incrementing a PDP-11 register from R7 prints out "R8" and the contents of R0.

The I/O page is in the address group 17XXXX.

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.

**LSI-11/23**

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 an 18-bit address and 18-bit data may be entered. Leading zeroes are assumed.

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 "R0" and the contents of R0.

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 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**

- i. A non-existent micro PC address

A carriage return (CR) is echoed and followed by just a line feed (LF).

No "H" command.

**LSI-11/23**

A carriage (CR) return is echoed and followed by another (CR) and line feed (LF).

"H" causes the LSI-11/23 to execute a microcode routine that, in effect, does nothing.



## APPENDIX E

# INTEGRATED CIRCUITS

### Bus Receivers and Bus Drivers

The equivalent circuits of LSI-11 bus-compatible drivers and receivers are shown in Figure 1. To perform the receiver and driver functions, Digital Equipment Corporation uses two monolithic integrated circuits with the characteristics listed in Table 1. A typical bus driver circuit is shown in Figure 2. Note that 8641 quad transceivers can be used, combining LSI-11 bus receiver and driver functions in a single package. Bus receiver (8640), bus driver (8881), and bus transceivers (8641) are shown in Figures 3, 4, and 5, respectively.

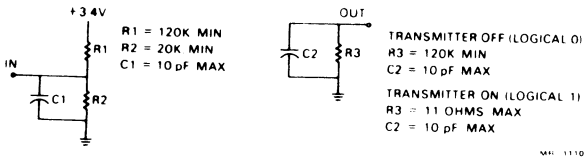


Figure 1 Bus Driver and Receiver Equivalent Circuits

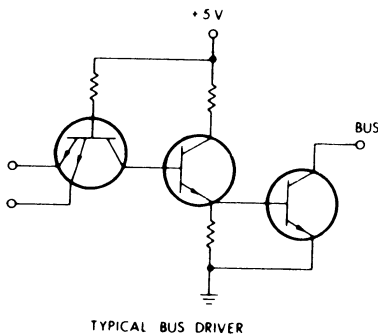
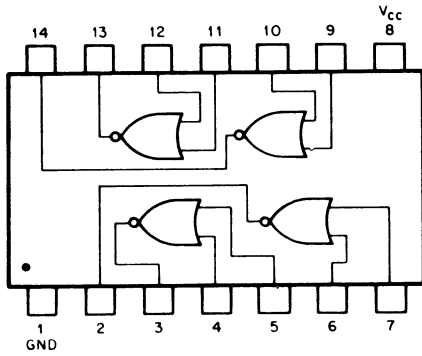
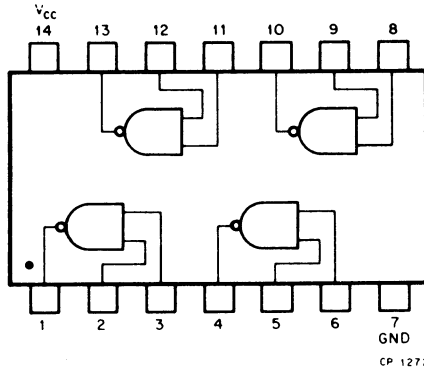


Figure 2 Typical Bus Driver Circuit

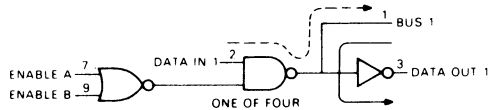
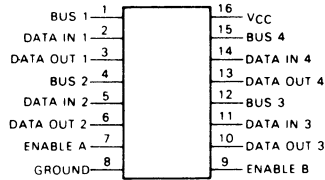
## Appendix E—Integrated Circuits



**Figure 3 8640 Quad 2-Input NOR Gates (Bus Receiver)**



**Figure 4 8881 Quad 2-Input NAND Gate (Bus Driver)**



**Figure 5 8641 Quad Unified Bus Transceiver (Bus Receiver/Driver)**

## Appendix E—Integrated Circuits

**Table 1 LSI-11 Bus Driver, Receiver, Transceiver Characteristics**

Device	Characteristic	Sym	Specifications	Notes	
Receiver (8640) (8641)	Input high voltage	$V_{IH}$	1.7 V min	1	
	Input low voltage	$V_{IL}$	1.3 V max	1	
	Input current at 3.8 V	$I_{IH}$	80 $\mu$ A max	1, 3	
	Input current at 0 V	$I_{IL}$	10 $\mu$ A max	1, 3	
	Output high voltage	$V_{OH}$	2.4 V min	2	
	Output high current	$V_{OH}$	(16 TTL loads)	2, 3	
	Output low voltage	$V_{OL}$	0.4 V max	2	
	Output low current	$I_{OL}$	(16 TTL loads)	2, 3	
	Propagation delay to high state	TPDH	10 ns min 35 ns max	4, 5	
	Propagation delay to low state	TPDL	10 ns min 35 ns max	1, 5	
	Driver (8881) (8641)	Input high voltage	$V_{IH}$	2.0 V min	
		Input low voltage	$V_{IL}$	0.8 V max	
		Input high current	$I_{IH}$	60 $\mu$ A max	6
Input low current		$I_{IL}$	-2.0 mA max	6	
Output low voltage 70 mA sink		$V_{OL}$	0.8 V max	1	
Output high leakage current at 3.5 V		$I_{OH}$	25 $\mu$ A max	1, 3	
Propagation delay to low state		TPDL	25 ns max	1, 5	
Propagation delay to high state		TPDH	35 ns max	1, 5	

### NOTES

1. This is a critical parameter for use on the I/O bus. All other parameters are shown for reference only.
2. This is equivalent to being capable of driving 16 unit loads of standard 7400 series TTL integrated circuits.
3. Current flow is defined as positive if into the terminal.
4. Conditions of load are 390  $\Omega$  to +5 V and 1.6 k $\Omega$  in parallel with 15 pF to ground for 10 ns min and 50 pF for 35 ns max.
5. Times are measured from 1.5 V level on input to 1.5 V level on output.
6. This is equivalent to 1.25 standard TTL unit loading of input.

Bus receivers and drivers should be well grounded and use  $V_{CC}$  to ground bypass capacitors. These gates should be located as close as practical to the module fingers which plug into the backplane and all etch runs to the bus should be kept as short as possible. Attention to these cautions should yield a module design with minimum bus loading (capacitance).



## APPENDIX F

### FALCON SBC-11/21 TIMING

The following fetch and execute times assume that the FALCON SBC-11/21 (KXT11) is transacting with local devices that do not require T-11 cycle slips when accessed.

Single Oper- and Instructions	Destination Mode	Fetch and Execute Time (usec)	Number of Bus Transac- tions	Number of Microcycles
CLR(B), COM(B),	0	2.44	1	4
INC(B), DEC(B),	1	4.27	3	7
NEG(B), ROR(B),	2	4.27	3	7
ROL(B), ASR(B),	3	5.49	4	9
ASL(B), SWAB,	4	4.88	3	8
ADC(B), SBC(B),	5	6.10	4	10
SXT, MFPS,	6	6.10	4	10
XOR	7	7.32	5	12
	0	2.44	1	4
	1	3.66	2	6
	2	3.66	2	6
TST(B)	3	5.49	3	8
	4	4.27	2	7
	5	5.49	3	9
	6	5.49	3	9
	7	6.71	4	11
	0	4.88	1	8
	1	6.10	2	10
	2	6.10	2	10
MTPS	3	7.32	3	12
	4	6.71	2	11



*Appendix F—SBC-11/21 FALCON Instruction Timing*

<b>Single Oper- and Instructions</b>	<b>Destination Mode</b>	<b>Fetch and Execute Time (usec)</b>	<b>Number of Bus Transac- tions</b>	<b>Number of Microcycles</b>
	5	7.93	3	13
	6	7.93	3	13
	7	9.16	4	15
<b>Double Op- erand Instructions</b>	<b>Source Mode</b>	<b>Source Mode Time (usec) Includes Fetch</b>		
MOV(B), CMP(B),	0	1.83	1	3
ADD, SUB,	1	3.05	2	5
BIT(B), BIC(B),	2	3.05	2	5
BIS(B),	3	4.27	3	7
	4	3.66	2	6
	5	4.88	3	8
	6	4.88	3	8
	7	6.10	4	10
<b>Double Op- erand Instructions</b>	<b>Destination Mode</b>	<b>Destination Mode Time (usec)</b>		
MOV(B), CMP(B),	0	0.61	0	1
ADD, SUB,	1	2.44	2	4
BIT(B), BIC(B),	2	2.44	2	4
BIS(B),	3	3.66	3	6
	4	3.05	2	5
	5	4.27	3	7
	6	4.27	3	7
	7	5.49	4	9
	0	0.61	0	1
	1	1.83	1	3
	2	1.83	1	3

*Appendix F—SBC-11/21 FALCON Instruction Timing*

<b>Double Op- erand Instructions</b>	<b>Destination Mode</b>	<b>Destination Mode Time (usec)</b>		
CMP(B), BIT(B)	3	3.05	2	5
	4	2.44	1	4
	5	3.66	2	6
	6	3.66	2	6
	7	4.88	3	8
<b>Jump and Subroutine Instructions</b>	<b>Destination Mode</b>	<b>Fetch and Execute Time (usec)</b>		
JMP	1	3.05	2	5
	2	3.66	2	6
	3	3.66	3	6
	4	3.66	2	6
	5	4.27	3	7
	6	4.27	3	7
	7	5.49	4	9
JSR	1	5.49	4	9
	2	6.10	4	10
	3	6.10	5	10
	4	6.10	4	10
	5	6.71	5	11
	6	6.71	5	11
	7	7.90	6	13
RTS	NA	4.27	2	7
SOB	NA	3.66	1	6
<b>Branch, Trap, and Interrupt Instructions</b>	<b>Destination Mode</b>	<b>Fetch and Execute Time (usec)</b>		
BR, BNE, BEQ, BPL, BMI, BVC,	NA	2.44	1	4

*Appendix F—SBC-11/21 FALCON Instruction Timing*

<b>Branch, Trap, and Interrupt Instructions</b>	<b>Destination Mode</b>	<b>Fetch and Execute Time (usec)</b>		
BVS, BCC, BCS,  BGE, BLT, BGT,  BLE, BHI, BLOS,  BHIS. BLO.				
EMT, TRAP, BPT, IOT	NA	9.77	7	16
RTI	NA	4.88	3	8
RTT	NA	6.71	3	11
<b>Miscellaneous and Condition Code Instructions</b>	<b>Destination Mode</b>	<b>Fetch and Execute Time (usec)</b>		
HALT	NA	8.54	5	14
WAIT	NA	2.44	1	4
				then loop
RESET	NA	22.28	1	39
NOP	NA	3.66	1	6
CLC, CLV, CLZ,  CLN, CCC, SEC,  SEV, SEZ, SEN,  SCC	NA	3.66	1	6
MFPT	NA	3.05	1	5

A processor cycle slip must occur if slow PROMs have been installed, or if the Q-Bus is accessed. (A slow PROM is one with an output enable time greater than 250 ns.) The FPLA output term CLSLIP L must be altered if the above referenced slow PROMS are used in the KXT11 sockets. A minimum of 1.2usec must be added to each bus transaction when this cycle slip occurs.

The measure of Q-BUS interrupt latency is the time from assertion of BIRQ until BIAKI is accepted by the interrupting device that is electrically closest to the processor on the Q-BUS.

The measure of local interrupt latency is the time from assertion of the request until the time T-11 is ready to fetch the first instruction in the interrupts' service routine. This is the time needed to push two words onto the stack and insert the new PL and PSW values for the interrupt.

DMA latency is known as the period of time between when a device asserts its BDMR and receives BDMGI when the device resides on the Q-BUS as the electrically closest DMA device to the processor.

Interrupt latency:	LOCAL	23.2usec
	Q-BUS	9.3usec

#### NOTE

The times listed for interrupt latency assume that the stack and vector memory reside on the KXT11-AA and that the Q-BUS device can assert BRPLY and vector within 600nsec after receiving IAKI. They also assume that the service latency depends on the

response time of the interrupting device i.e., RDIN to TRPLY and negation of TRPLY. Service latency is the time from BIRQ assertion until the T-11 is ready to fetch the first instruction in the interrupt's service routine.

DMA latency:	1.3usec minimum	11.0usec maximum
WAIT instruction latencies:		
	internal vector	11.8usec
	external vector	12.4usec
	DMR	5.06usec



## APPENDIX G

### DIGITAL MICROCOMPUTER DOCUMENTATION

DIGITAL offers many microcomputer-related product technical guides, summaries, bulletins, and brochures that are very useful as supplementary material to this Handbook. A list of these publications follows:

<b>Catalog Number</b>	<b>Publication</b>
EK-11V03-JP	PDP-11/V03 Unit Assembly IPB
EK-11V03-TM	PDP-11/V03 System Manual
EK-11V23-IP	PDP-11/V23 Unit Assembly IPB
EK-11V23-OP	PDP-11/V23 System Manual
EK-1V03L-IP	PDP-11/V03-L Unit Assembly IPB
EK-AXV11-UG	LSI-11 Analog System User Guide
EK-BA11A-IP	BA11-A Unit Assembly IPB
EK-BA11A-TM	BA11-A Mounting Box & Power Tech
EK-BA11F-IP	BA11-F Mounting Box IPB
EK-BA11K-IP	BA11-K Mounting Box IPB
EK-BA11K-OP	BA11-K Mounting Box User Manual
EK-BA11K-TM	BA11-K Mounting Box Technical Manual
EK-BA11L-IP	BA11-L Mounting Box IPB
EK-BA11L-TM	BA11-L Technical Manual
EK-BA11M-IP	BA11-M Mounting Box IPB
EK-BA11N-IP	BA11-N Mounting Box IPB
EK-BA11N-TM	BA11-N Mounting Box Technical Manual
EK-BA11N-UG	BA11-N Mounting Box User Manual

*Appendix G—LSI-11 Documentation*

<b>Catalog Number</b>	<b>Publication</b>
EK-BA11P-IP	BA11-P Unit Assembly IPB
EK-BA11U-IP	BA11-U Mounting Box IPB
EK-BA11V-IP	BA11-V Box Assembly IPB
EK-BA11V-RG	BA11-VA Configuration Guide
EK-BA1KP-IN	BA11-KP Installation Manual
EK-BAM11-TM	BAM11 Technical Manual
EK-BAM11-UG	BAM11 Status Alarm Monitor User Guide
EK-BDV11-TM	BDV11 Technical Manual
EK-DLV11-OP	DLV11-E/F User Manual
EK-DZV11-TM	DZV11 Asynch Multi Technical Manual
EK-DZV11-UG	DZV11 Asynch MTLPRX Guide & Addendum
EK-FPF11-TM	FPF11 Floating-Point Processor
EK-H927A-CG	H9275-A Backplane Configura- tion Sheet
EK-KD1HA-CG	LSI-11/2 Processor Configura- tion Sheet
EK-KDF11-UG	KDF11-AA User Guide
EK-KDFAA-CG	LSI-11/23 Processor Configura- tion Sheet
EK-KUV11-TM	LSI-11 WCS User Guide
EK-LSI11-MC	LSI-11 PDP-11/03 Maintenance Card
EK-LSI11-TM	LSI-11 PDP-11/03 User Manual
EK-LSIFS-SV	LSI-11 System Service Manual
EK-MCV1D-UG	MCV11-D User Guide
EK-MSV1D-OP	MSV11-D/E User Manual
EK-MSVOP-UG	MSV11-P User Guide
EK-ORL01-IP	RL01 Disk Drive IPB

*Appendix G—LSI-11 Documentation*

<b>Catalog Number</b>	<b>Publication</b>
EK-ORL02-IP	RL02 Disk Drive IPB
EK-ORX01-IP	RX01 Floppy Disk IPB
EK-ORX01-MM	RX01/08/11 Maintenance Manual
EK-ORX02-IP	RX02 Floppy Disk IPB
EK-ORX02-TM	RX02 Floppy Disk SYS Technical Manual
EK-ORX02-UG	RX02 Floppy Disk System User Guide
EK-OSB11-DG	SB11 Series OEM SYS Designer Guide
EK-OSB11-IP	SB11 Microcomputer IPB
EK-OTU58-CG	TU58-VA DECtape II Configuration
EK-OTU58-EC	TU58 DECtape Customer Equipment Care
EK-OTU58-IP	TU58 Cartridge Tape Drive IPT
EK-OTU58-PS	TU58 DECtape II Pocket Service Guide
EK-OTU58-TM	TU58 DECtape-11 Technical Manual
EK-OTU58-UG	TU58 DECtape-11 User Guide
EK-OTU58-WS	TU58 DECtape Installation Sheet
EK-RL012-PG	RKL01/02 Pocket Service Guide
EK-RL012-TM	RL01/02 Disk Drive Tech Manual
EK-RL012-UG	RL01/02 User Guide
EK-RL012-WS	RL01/02 P.M. Worksheet 25/PKG
EK-RLTER-PS	RL01/02 Pocket Service Guide
EK-T03LO-OP	PDP-11/T03-L System Manual



*Appendix G—LSI-11 Documentation*

<b>Catalog Number</b>	<b>Publication</b>
EK-V03LO-OP	PDP-11/V03-L System Manual
ED-18321-53	Asynchronous Interface LSI-11 (DLV11-E)
AA-K724A-TC	BASIC-11/RT-11 Installation Guide & Release Notes
ED-09371-53	Battery Backup-LSI-11 Eng. Note
EB 19187-75	Cables Handbook
EA-19971-18	Coming to Terms with IBM & Networking
GA-18195-75	Computer Maintenance Alternative US
ED-06703-92	DDV11-B 9X6 Slot LSI-11 Backplane Bulletin
EA-18097-18	Digital Family of Terminals Brochure
EJ-19822-53	DPV11-DA Synchronous Line Interface
ED-20086-53	DPV11-DA Option Bulletin
ED-18511-53	DRV11-J Option Bulletin
ED-18326-53	EPROM/PROM/ROM Module LSI-11 (MRV11-C)
ED-17472-53	EPROM/PROM/ROM Module March 79
ED-18322-53	Four-Channel LSI-11 Micro (DLV11-J)
EA-20360-53	H9275-A Option Bulletin
ED-18762-53	High Density Parallel Interface DRV11-J
ED-07494-53	IBV11-A Interface Option for LSI-11
ED-18328-53	IEEE Interface OPT LSI-11 (IBV11-A)
ED-18967-18	LA38 DECwriter IV Printing Terminal
ED-18966-18	LA120 DECwriter III Data Sheet
ED-19211-56	LA120-RA DECprinter III Bulletin 4/80
ED-08193-18	LA180 DECprinter Data Sheet 06/78
ED-17474-53	LSI-11 Mounting Chassis/Power March 79
ED-18327-53	LSI-11 Multifunction Module (MXV11)

*Appendix G—LSI-11 Documentation*

<b>Catalog Number</b>	<b>Publication</b>
ED-18324-53	LSI-11 Option Memory Module (MSV11-DD)
ED-18647-18	LSI-11 The Complete Family Brochure
ED-18323-53	LSI-11/2 Central Processor (KD11-HA)
ED-21361-20	LSI-11/23 Data Sheet July 81
ED-18325-53	LSI-11/23 High Performance (KDF11-AA)
EH-17898-20	LSI-11/23 (PDP-11/23) Reference Card
EA-17057-18	LSI-11/23-S Microcomputer
EE-19213-53	Micro Products Group Product Description
EB-18451-20	Microcomputer and Memories Handbook
EB-20175-20	Microcomputer Interface Handbook 1980
EJ-18382-53	Microcomputer Products Group U-Note
EA-18334-53	Microcomputer Products Selection Guide
EA-20065-53	Microcomputer Software Brochure
ED-17470-53	MXV11 Multifunction Module March 79
EA-18924-18	PDP-11 Microcomputer Members Brochure
EH-07043-53	PDP-11/03, LSI-11 Reference Card
EE-20305-53	Price List (For Internal Distribution Only!!!)
EA-19442-18	RT-11 Single-User Realtime Systems
EJ-18922-28	RT-11 Technical Summary
AE-3438C-TC	SPD 9.2.2 Papertape Support Package
AE-J514A-TC	SPD 10.16.0 RT-11
AE-D431F-TC	SPD 10.72.6 DECNET-RT
AE-3393M-TC	SPD 12.1.14 RT-11
AE-D007D-TC	SPD 12.4.3 RT2
AE-3394H-TC	SPD 12.5.8 BSC-11/RT-11
AE-3395L-TC	SPD 12.10.12 F IV/RT-11
AE-H257B-TC	SPD 12.14.1 Instrument Bus Sub
AE-3397F-TC	SPD 12.20.6 MU BSC-11/RT-11

<b>Catalog Number</b>	<b>Publication</b>
AE-H585B-TC	SPD 12.21.1 PROM/RT-11
AE-H509C-TC	SPD 12.22.2 FMS-11/RT-11
AE-J673A-TC	SPD 12.29.0 Fortran/RT-11 Lab Extensions
AE-J698A-TC	SPD 14.42.0 APL-11/RT
AE-D607C-TC	SPD 15.44.2 LSP-11
AE-3413E-TC	SPD 15.45.6 SSP-11
AE-D370C-TC	SPD 15.90.2 LSI-11 Micro Tools
EA-18784-53	The Story Behind LSI-11 Microcomputer
EA-19973-53	TU-58 Option Bulletin October 1980
ED-17473-53	Universal Prom Programmer March 79
ED-19000-53	VT103/LSI-11 Video Terminal April 1980

### **TECHNICAL GUIDES/MANUALS**

EK-BA11N-TM	BA11-N Mounting Box Tech Manual
EK-BDV11N-TM	BDV11 Technical Manual
EK-01387-92	CHIPKIT User Guide
EK-DLVKA-IN	DLV11-KA Installation Manual
EK-DPV11-UG	DPV-11 Synchronous Interface User Guide
EK-DPV11-TM	DPV-11 Technical Manual
EK-DRV1B-OP	DRV11-B Interface User Manual
EK-DRV1J-UG	DRV11-J Interface User Manual
EK-DRV11-OP	DRV11-P Foundation Module User Manual
EK-DUV11-TM	DUV-11 Line Interface Technical Manual
EK-DUV11-OP	DUV11 User Guide
EK-DZV11-TM	DZV11 Asynch Multi Technical Manual
EK-H927A-CG	H9275-A Configuration Guide
EK-IBV11-UG	IBV11-A User Manual
EK-LA120-TM	LA120 Technical Manual

## Appendix G—LSI-11 Documentation

<b>Catalog Number</b>	<b>Publication</b>
EK-KDFAA-CG	LSI-11/23 Processor Configuration Sheet
EK-PWRPK-CL	Power and Packing Catalog
EK-RL012-TM	RL01/02 Disk Drive Technical Manual
EK-RLV11-TD	RLV11 Controller Tech Description Manual
EK-ORX02-TM	RX02 Floppy Disk Systems Technical Manual
EK-OTU58-TM	TU58 DECtape-11 Technical Manual
EK-VT100-UG	VT100 User Guide
ED-VT103-CG	VT103 Configuration Guide
EK-VT103-UG	VT103 LSI-11 User Guide and Addendum
EK-VT103-IP	VT103 Unit Assembly IPB
EK-VT1X3-CG	VT1X3-MM Maintenance Module Configuration Sheet

### **MISCELLANEOUS**

	BDV11 Bootstrap & Diagnostic
	H9281 Backplane Engineering Spec
	Technical OEM Product Kit
EB-19402-20	PDP-11 Processor Handbook



## APPENDIX H

### PDP-11/23-PLUS TIMING

The PDP-11/23-PLUS (KDF11-B) CPU executes PDP-11 instructions as a series of microcode cycles. A data fetch consists of an address cycle and a bus DIN cycle. A data write consists of an address cycle and a bus DOUT cycle. An instruction fetch consists of an address cycle, a bus DIN cycle, and a non-I/O cycle. The execution of an instruction typically consists of one or more non-I/O cycles. Floating Point instructions also include interchip DIN and DOUT cycles which move data between the DATA and MMU chips. The execution time for an instruction depends on the type of instruction, modes of addressing used, the type of memory referenced and whether the memory management unit is enabled or disabled.

Each microcode cycle consists of an integral number of clock pulses which occur at 75 ns intervals. The number of clock pulses and time required to complete the most common microcode cycles are listed in Table 1. The time required for bus DIN or DOUT microcode cycles which access either the memory management registers (MMU DIN or DOUT) or the KDF11-B on-board peripherals (IDAL bus DIN or DOUT) are listed in Table 2. The KDF11-B peripherals include the bootstrap and diagnostic ROMS, the line clock logic and the serial line units.

**Table 1 KDF11-B Common Microcode Cycle Times**

Type of Cycle	Clock Pulses	Time (ns)
Address (No Relocation)	5	375
Address (Relocation)	8	600
LSI-11 Bus DIN (1)	10	750
LSI-11 Bus DIN (2)	11	825
LSI-11 Bus DOUT (3)	11	825
INTERCHIP DIN	5	375
INTERCHIP DOUT	5	375
Non I/O (Micro-NOP)	4	300

The KDF11-B detects RRPLY assertion within 112.5 ns of the time it asserts TDIN. (Typical for peripherals which assert TRPLY as soon as they receive RDIN asserted.)

The KDF11-B detects RRPLY assertion within 337.5 ns of the time it asserts TSYNC. (Typical for the MSV11-P Parity Memory).

The KDF11-B detects RRPLY assertion within 150 ns of the time it asserts TDOUT. (Typical for peripherals and memories which assert TRPLY as soon as they receive RDOUT asserted. This includes the MSV11-P Parity Memory.)

**Table 2 KDF11-B Peripheral Microcode Cycle Times**

Type of Cycle	Clock Pulses	Time (ns)
IDAL Bus DIN	8	600
IDAL Bus DOUT	8	600
MMU DIN	7	525
MMU DOUT	7	525

### BASIC INSTRUCTION TIMING

This system lists the source, destination and fetch/execute times for the KDF11-B Basic Instruction Set. KDF11-B instruction times are calculated using the following equation:

$$\text{INSTRUCTION TIME} = \text{Basic Time} + \text{Source Time} + \text{Destination Time}$$

$$(\text{Basic Time} = \text{Fetch Time} + \text{Execute Time})$$

The basic, source and destination times were calculated from the microcode cycle times listed in Table 1. LSI-11 Bus DIN (2) and DOUT (3) times of 825 ns were used for the MSV11-P Parity Memory which has the specifications listed in Table 3.

**Table 3 MSV11-P Parity Memory**

Bus Cycles	Access Time (ns)		Cycle Time (ns)	
	Typ	Max	Typ	Max
DATI	240	260	560	590
DATO (B)	90	120	610	640
DATIO (B)	660	690	1175	1210

The instruction execution times for systems with memory management enabled or disabled are listed in Tables 4 through 7.

**Table 4 Source Address Times**

Instruction	Source Mode	Memory Cycles	Time (microseconds) with Memory Management	
			Enabled	Disabled
	0	0	0	0
ADD, SUB	1	1	1.425	1.200
MOV(B), CMP(B)		2	1	1.425
			1.200	
BIS(B), BIC(B)		3	2	2.850
			2.400	
BIT(B)	4	1	1.725	1.500
	5	2	3.150	2.700
	6	2	3.150	2.700
	7	3	4.575	3.900
	0	0	1.275	1.275
MUL, DIV	1	1	1.725	1.450
ASH, ASHC	2	1	1.725	1.450
MFPI, MFPD	3	2	2.850	2.400
MTPS	4	1	1.725	1.500
	5	2	3.150	2.700
	6	2	3.150	2.700
	7	3	4.575	3.900

**Table 5 Destination Address Times**

Instruction	Source Mode	Memory Cycles	Time (microseconds) with Memory Management	
			Enabled	Disabled
MOV(B), CLR(B)		0	0	0
	0			
SXT, MFPS	1	1	2.025	1.800
MTP1, MTPD	2	1	2.025	1.800
	3	2	3.150	2.700
	4	1	2.025	1.950
	5	2	3.450	3.000
	6	2	3.450	3.000
	7	3	4.875	4.500



Instruction	Source Mode	Memory Cycles	Time (microseconds) with Memory Management	
			Enabled	Disabled
CMP(B), BIT(B) 0		0	0	0
TST(B)	1	1	1.725	1.500
	2	1	1.725	1.500
	3	2	2.850	2.400
	4	1	1.725	1.500
	5	2	3.150	2.700
	6	2	3.150	2.700
	7	3	4.575	3.900
ADD, SUB	0	0	0	0
INC(B), DEC(B) 2.625		1	1	2.850
COM(B), NEG(B) 2.625		2	1	2.850
ROR(B), ROL(B) 3.825		3	2	4.275
ASR(B), ASL(B) 2.625		4	1	2.850
BIS(B), BIC(B) 4.125		5	2	4.575
ADC, SBC	6	2	4.575	4.125
XOR, SWAB	7	3	6.000	5.325

**Table 6 Basic (Fetch & Execute) Time**

Instruction	Memory Cycles	Time (microseconds) with Memory Management	
		Enabled	Disabled
MOU, CMP, BIT, BIC, BIS, ADD, SUB, SXT, CLR, TST, COM, INC, DEC, NEG, ADC, SBC, RCR, ROL, ASR, ASL, SWAB, MFPS	1	2.025	1.800

Instruction	Memory Cycles	Time (microseconds) with Memory Management	
		Enabled	Disabled
MTPS	1	3.600	3.375
MFPI, MFPD	2	4.050	3.600
MTPI, MTPD	2	4.725	4.275
SCB (NO BRANCH) 2.400		1	2.625
SOB (BRANCH)	1	2.925	2.700
ALL BRANCH	1	2.025	1.800
CLN, CLE, CLV, CLC, SEN, SEZ, SEV, SEC, CCC, SCC	1	2.925	2.700
RTS	2	3.750	3.300
MARK	2	5.325	4.875
RTI	3	6.225	5.550
RTT	3	7.500	6.825
IOT, BPT	5	10.500	9.375
EMT, TRAP	5	9.525	8.850
WAIT	1	3.375	3.150
MUL	1	33.300	33.075
DIV	1	49.650	49.425
ASH	1	24.825	24.600
ASHC	1	46.050	45.825

**NOTE**

1. The instruction times for MUL, DIV, ASH, and ASHC are operand dependent and could be less than the values given above.
2. The instruction times for the RESET and Halt instructions are Mode/Option Dependent.

**Table 7 Jump Instruction Times**

Instruction	Dest. Mode	Memory Cycles	Time (microseconds) with Memory Management	
			Enabled	Disabled
JMP	1	1	2.325	2.100
	2	1	2.625	2.400
	3	2	3.450	3.000
	4	1	2.625	2.400
	5	2	3.750	3.300
	6	2	3.750	3.300
	7	3	5.175	4.500
JSR	1	2	4.350	3.900
	2	2	4.650	4.200
	3	3	5.475	4.800
	4	2	4.650	4.200
	5	3	5.775	5.100
	6	3	5.775	5.100
	7	4	7.200	6.300

**DMA AND INTERRUPT LATENCIES**

DMA latency is the time required for the first DMA device to obtain bus mastership after it asserts a direct memory access request (BDMR L). The DMA latency is 1.35 microseconds, maximum. The maximum DMA latency was calculated for a relocated address cycle followed by a DOUT cycle. The processor disables DMA grant (BDMGO L) from the end of the address cycle phase time until four 75 ns intervals after the DOUT cycle phase time.

Interrupts (BR requests) are acknowledged by the processor at the end of the current instruction. Interrupt latency is defined as the time required by the KDF11-B to assert an interrupt acknowledge (BIAKO L) after receiving an interrupt request. Interrupt service time is defined as the time required to fetch the first service routine instruction after asserting BIAKO L. The interrupt latency time and the interrupt service time must be added to obtain the total time from the reception of the interrupt request to the fetch of the first service routine instruction. The specifications for interrupt latency and interrupt service times are as follows.

Interrupt Latency            –5.475 s, typ [MOV X(R7),R0]  
                                   12.600 –s, max (except EIS)  
                                   54.225 –s, max (including EIS)

*Appendix H—PDP-11/23-PLUS Instruction Timing*

Interrupt Service      8.625  $\mu$ s (Memory Management Off)  
                                 9.750  $\mu$ s (Memory Management On)

**NOTE**

1. Interrupt and DMA latencies assume a KDF11-B with memory management enabled and using MSV11-P memory.
2. The maximum interrupt latencies were calculated for ADD @X(R7), @Y(R7) and for DIV @X(R7).



## INDEX

- aborts, 272
  - status registers and, 282
- ABSD instruction, 127
- ABSF instruction, 127
- absolute mode addressing, 187
- access control field (ACF), 276-277
- access control keys, 272
- accumulators, 43
  - in floating point option, 123
- accuracy, in floating point, 123-125
- ACF (access control field), 276-277
- active page field (APF), 280, 281
- active page register (APR), 266, 268, 269, 281
- ADCB instruction, 78
- ADC instruction, 128-129
- ADDD instruction, 128-129
- ADDF instruction, 79
- addresses
  - in FALCON SBC-11/21 microcomputer, 307, 310-314
  - memory relocation of, 267-268
  - ODT specification of, 170
  - reserved for control status register, 509-511
  - vector, 197
  - virtual and physical, 279-281
  - see *also* starting addresses
- addressing, 36, 266
  - of devices, by buses, 225
  - direct address mode (in MRC11-C memory) for, 435-439
  - of floating point option instructions, 123
  - in LSI-11 instruction set, 65
  - in MCV11-D memory, 535
  - of memory and peripherals, 32-33
  - in MRV11-AA memory, 412-414
  - in MRV11-BA memory, 480
  - in MSV11-CD memory, 480
  - in MSV11-D, E memory, 489, 496
  - in MSV11-L memory, 516-518
  - in MSV11-P memory, 650-655
  - in MXV11-A memory and asynchronous serial line interface, 575, 578
  - in PROMs, 464, 467, 469, 470
  - window mapping mode (in MRV11-C memory) for, 440-444
- addressing modes, 3-4, 36, 43-62
  - position-independent code, and 185-186
- address jumpers
  - for MRV11-AA memory, 406-410
  - for MSV11-B memory, 476
  - for MSV11-A memory and asynchronous serial line interface, 578
- address latch, 537-538
- AIO (address-input-output) codes, 335-336
- APF (active page field), 280, 281
- APL-11 (language\_\_\_, 7
- applications, 8-10
- APR, see active page register
- architecture
  - of FALCON SBC-11/21 microcomputer, 36
  - of LSI-11 microcomputer, 29-35
  - of LSI-11/2 microcomputer, 37-38
  - of LSI-11/23 microcomputer, 36-37
  - of PDP-11/23-PLUS microcomputer, 607
  - stacks in, 189

**ASCII (American Standard Code for Information Interchange),**  
 conversion routines for, 217  
**ASHC instruction,** 110-111  
**ASH instruction,** 110  
**ASLB instruction,** 79  
**ASL instruction,** 79  
**ASRB instruction,** 80  
**ASR instruction,** 80  
**assembly language**  
     conversion routines for, 213  
     windowed programs written in, 455  
**asynchronous serial line interface (MXV11-A),** 541-580  
     @ (at sign), 45  
     @ (at sign) ODT command, 174  
**autodecrement deferred mode,** 50-51  
**autodecrement mode,** 49-50  
     for stack addressing, 190  
**autoincrement deferred mode,** 49  
**autoincrement mode,** 48  
**BA11-ME expansion enclosures,** 39  
**BA11-NE expansion enclosures,** 40  
**BA11-N processor boxes,** 602  
**BA11-S expander boxes,** 601  
**BA11-VA mounting boxes,** 39  
**ODT command,** 174  
**backplane jumpers,** 635-639  
**backplanes,** 39  
     installation of FPF11 floating-point processor in, 288  
     LSI-11 Bus wiring and, 247-248  
     MSV11-D, E memories and, 503-504  
     multiple, configuration of, 249-250  
     options with, 31  
**BANK 7 decoder,** 365-366  
**BANK 7 SEL H signal,** 496  
**BASIC-11 (language),** 7  
**BASIC-PLUS-2 (language),** 7  
**battery backup**  
     for MCV11-D memory, 535-536  
     for MSV11-CD memory, 483-484  
     for MSV11-D, E memories, 485, 489-490  
     for MSV11-L memory, 511  
**baud rate generator,** 580, 628  
**baud rate jumpers,** 565-567  
**baud rates,** 617, 628-629  
**BBS7 L signal,** 225, 556, 575, 652  
**BC02D-03 jumper/cable terminator assembly,** 602  
**BC20M-50 cable,** 567  
**BCC instruction,** 80  
**BCS instruction,** 80-81  
**BDAL (bus data address lines),** 346, 504, 505, 537  
**BDAL bus drivers,** 225, 226, 230, 241, 373  
**BDAL L signal,** 386  
**BDCOK H signal**  
     in LSI-11 Bus, 243, 244, 250  
     in LSI-11/2 microcomputer, 367, 373  
     in LSI-11/23 microcomputer, 345, 346  
**BDIN L signal**  
     in LSI-11 Bus, 225, 226, 240, 241  
     in LSI-11/2 microcomputer, 364, 368-370  
     in MRV11-BA memory, 426, 429  
     in MRV11-P memory, 655  
     in MXV11-A memory and asynchronous serial line interface, 577, 580  
**BDMGI L signal,** 234, 235  
**BDMGO L signal,** 234, 340-344  
**BDMG signals,** 220

**BDMR L signals**, 234, 340, 343  
**BDCOK H signal**, 538  
**BDOUT L signal**  
   in LSI-11 Bus, 230  
   in LSI-11/2 microcomputer, 364  
   in MRV11-BA memory, 429  
   in MRV11-C memory, 450  
   in MSV11-L memory, 529  
   in MSV11-P memory, 652  
   in MXV11-A memory and asynchronous serial line interface, 577  
**BDV11 clock status/control register**, 567  
**BDV11 module**, 603  
**BEQ instruction**, 81  
**BEVNT L signal**  
   in LSI-11 microcomputer, 385  
   in LSI-11/2 microcomputer, 360, 370  
   in MXV11-A memory and asynchronous serial line interface, 567, 572  
**BGE instruction**, 81-82  
**BGT instruction**, 82  
**BHALT L signal**, 164  
   in LSI-11 Bus, 243  
   in LSI-11 microcomputer, 383-385  
   in LSI-11/2 microcomputer, 357-360, 368  
   in LSI-11/23 microcomputer, 325  
   in PDP-11/23-PLUS microcomputer, 625  
**BHI instruction**, 82-83  
**BHIS instruction**, 83  
**BIACK signals**, 220  
**BIAKI L signal**, 241  
**BIAK L signal**, 369, 370  
**BIAKO L signal**  
   in LSI-11 Bus, 240, 241  
   in LSI-11/2 microcomputer, 369  
**BICB instruction**, 83  
**BIC instruction**, 83  
   binary radix points, 154  
**BINIT L signal**, 243, 244, 447, 580  
**BIRQ4 L signal**, 238  
**BIRQ L signal**, 369, 370  
**BISB instruction**, 83-84  
**BIS instruction**, 83-84  
**BITB instruction**, 84  
**BIT instruction**, 84  
**BLE instruction**, 84-85  
**block number (BN)**, 280, 281  
**blocks**, 269  
**BLO instruction**, 85  
**BLOS instruction**, 85  
**BLT instruction**, 86  
**BMI instruction**, 86  
**BN (block number)**, 280, 281  
**BNE instruction**, 87  
**boards (LSI-11)**, 39  
**Board Selection Decode Logic**, 649-650  
**boot and diagnostic ROM jumpers**, 634-635  
**boot/diagnostic switches and jumpers**, 632, 634-635  
**Boot/Diagnostic switch units**, 622, 632  
**bootstrap address decoder**, 578  
**Bootstrap Check-Out Procedure**, 393-395  
**bootstrap/diagnostic configuration switches**, 632-634  
**bootstrap and diagnostic registers**, 613-617  
**bootstrap jumpers**, 563  
**bootstrap loader ODT command**, 178-179  
**bootstrapping**  
   on MRV11-C memory, 444, 459



- in MXV11-A memory and asynchronous serial line interface, 554, 555, 563-565 or PDP-11/23-PLUS microcomputers, 610-612 of PROMs, 463
- boxes, 39-40, 601, 602
- BPL instruction, 87
- BPOK H signal
  - in LSI-11 Bus, 243, 244, 250
  - in LSI-11/2 microcomputer, 366, 368
  - in LSI-11/23 microcomputer, 329
  - in PDP-11/23-PLUS microcomputer, 625
- BPT instruction, 87-88
- branch instructions, 65, 68, 74-75
- break halt jumpers, 630
- BREAK key, 391, 566
- break signals, 566, 617
- BREF L signal, 243, 386
- BR instruction, 88
- BRPLY bus driver, 580
- BRPLY L signal
  - in LSI-11 Bus, 225, 226, 230, 241
  - in LSI-11/2 microcomputer, 364, 368-370
  - in LSI-11/23 microcomputer, 339
  - in MRV11-BA memory, 426, 429
  - in MSV11-B memory, 476
  - in MSV11-CD memory, 483
  - in MSV11-D, E memories, 498, 505
  - in MXV11-A memory and asynchronous serial line interface, 575, 577
- BS7 H signal, 575
- BSACK L signal
  - in LSI-11 Bus, 234, 235
  - in LSI-11/23 microcomputer, 342, 343
- BSYNC L signal
  - in direct memory access, 235
  - in LSI-11 Bus, 224-226, 230
  - in LSI-11/2 microcomputer, 364, 370
  - in LSI-11/23 microcomputer, 339-340
  - in MCV11-D memory, 537
  - in MRV11-BA memory, 425, 428, 429
  - in MSV11-L memory, 528, 529
  - in MSV11-P memory, 652, 655
  - in MSV11-A memory and asynchronous serial line interface, 575, 578, 580
- buffers, stack use and, 194
- bus address interface, 459
- bus arbitration, 220
- bus arbitration, 29
- BUS CYC H signal, 336
- bus cycle protocol, 220, 224-225
- bus cycles, 31, 32
- BUS CYC L signal, 343
- bus data address lines, *see* BDAL
- bus driver enable logic, 373
- bus drivers, 245-246
- buses
  - data-address lines, 334-335
  - errors in, 368-369
  - LSI-11 Bus, 4, 29-31, 219-262
  - TDAL, 316
- bus grant continuity, 484
- bus grant continuity jumpers, 649
- BUS INIT(L) signal, 527, 661
- bus interfaces, 361-366
- bus masters, 29, 220, 224, 225
- bus mastership acquisition phase, 234
- bus mastership relinquish phase, 235
- bus receivers, 246
- bus restrictions
  - on MRV11-AA memory, 415
  - on MXV11-A memory and asynchronous serial line interface, 580

**Bus Sync Logic**, 339  
**bus termination**, 246-247  
**bus transceivers**, 424  
**BVC instruction**, 88  
**BVS instruction**, 89  
**BWTBT L signal**  
   in LSI-11 Bus, 225, 229, 230  
   in LSI-11/2 microcomputer, 364  
   in MSV11-D, E memories, 502, 503  
   in MSV11-L memory, 528  
   in MSV11-P memory, 652  
   in MXV11-A memory and asynchronous serial line interface, 575, 577  
**byte addressing**, 33,65  
**byte instructions**, 68  
**byte/word select function**, 538  
**cables**, 567-569  
**CALLWO**, 455-456  
**card guides**, 39  
**carriage return ODT command**, 166-167, 172-173  
**CAS (column address strobes)**, 518,525, 575  
**C bit**, 71-72  
**CCC instruction**, 90  
**CDAL (Chip Data and Address Line) bus** 607  
**central processing units (CPUs)**, *see processors*  
**CFF instruction**, 129  
**character formats**, 629-632  
**charge pump circuits**  
   in LSI-11/2 microcomputer, 372  
   in MRV11-BA memory, 430-431  
   in MSV11-D, E memories, 503-504  
   in MSV11-P memory, 660  
   in MXV11-A memory and asynchronous serial line interface, 578  
   checkpointing, using RSX-11M, 6  
**CHIP CLK**, 345  
**Chip Data and Address Line (CDAL) bus**, 607  
   chip-enable function, 458  
**CHIP RESET**, 349-350  
   chip select (CSEL), 331-332  
   chip sets  
     addressing of, in MRV11-C memory, 439  
     of LSI-11/2 microcomputer, 37, 360-371  
   chopping (truncating), 118, 124  
**C instruction**, 89-90  
**CIS (Commercial Instruction Set)**, 584, 607, 609  
**CLC instruction**, 90  
**CLN instruction**, 90  
**clock pulse circuits**, 372  
**clocks**  
   in FPF11 floating-point processor, 287, 295-296  
   Line Time Clock Program for, 398-399  
   in LSI-11/2 microcomputer, 372  
   in LSI-11/23 microcomputer, 324, 336, 345-350  
   in MXV11-A memory and asynchronous serial line interface, 567  
   in PDP-11/23-PLUS microcomputers, 609, 617, 628  
   clock stop cycle, 347-348  
   clock stutter cycle, 347  
**CLRB instruction**, 89  
**CLRD instruction**, 129-130  
**CLRF instruction**, 129-130  
**CLR instruction**, 89  
**CLV instruction**, 90-91  
**CLZ instruction**, 91  
**CMPB instruction**, 91

**CMPD instruction**, 130-131  
**CMPF instruction**, 130-131  
**CMP instruction**, 91  
column address strobes  
(CAS), 518, 525, 575  
**COMB instruction**, 92  
**COM instruction**, 92  
**commands**  
for console octal debugging  
technique operations, 161-179  
supported on PB11 universal  
PROM programmer option, 473  
**Commercial Instruction Set**  
(CIS), 584, 607, 609  
**Commercial packaged**  
**systems**, 590  
**communications, master-**  
**slave**, 29  
**comparators**, 440  
**condition-code bits**, 61-72  
**condition-code instructions**, 65,  
71-73  
**condition codes**, 35  
for FIS floating point, 155  
in LSI-11/23 microcomputer, 333  
**configuration register**, 615  
**configurations**  
of FALCON SBC-11/21  
microcomputers, 302-314  
of FPF11 floating-point  
processor, 288-291  
of LSI-11 Bus, 248-250  
of LSI-11 microcomputers, 379-  
386  
of LSI-11/2 microcomputers, 355-  
360  
of LSI-11/23 four-level  
interrupts, 241-242  
of LSI-11/23  
microcomputers, 323-330  
of MCV11-D memory, 533-536  
of MRV11-AA memory, 405-412  
of MRV11-BA memory, 417-421  
of MRV11-C memory, 433-455  
of MSV11-B memory, 475-478  
of MSV11-CD memory, 480-484  
of MSV11-D, E memories, 488- 490  
of MSV11-L memory, 509-516  
of MSV11-P memory, 644-649  
of MXV11-A memory and  
asynchronous serial line  
interface, 543-563  
of PDP-11/12-PLUS  
microcomputers, 622-639  
*see also* jumper configurations  
**console ODT**, 161-165  
**consoles**  
Check-Out Procedure for, 391  
octal debugging technique  
commands from, 161-179  
**console serial line units**, 617-618,  
628-632  
**control chips**  
on LSI-11/2 microcomputer, 360,  
361  
on LSI-11/23 microcomputer, 36-  
37, 331-332  
**control functions, in LSI-11**  
**Bus**, 342-344  
**control logic**  
in FPF11 floating-point  
processor, 295-296  
in MCV11-D memory, 538  
in MRV11-BA memory, 423-429  
in MSV11-D, E memories, 522-523  
in MSV11-P memory, 656  
in MXV11-A memory and  
asynchronous serial line  
interface, 575  
**control-shift-S ODT**  
**command**, 169, 179  
**Control and Status Registers**  
(CSRs)  
in MRV11-C memory, 440-442,  
447, 449  
in MSV11-L memory, 509-511,  
518, 522, 526-529  
in MSV11-P memory, 647-648,  
652, 658, 660-662

- in MXV11-A memory and asynchronous serial line interface, 558
- control store (in FPF11), 293-295
- conversion routines, 213-217
- COPY command, 473
- coroutines, 203-207
- CPU option jumpers, 624
- CPUs, *see* processors
- CSEI L signal, 295
- CSEL (chip select), 331-332
- CSR address logic, 652
- CSR Read operations, 655
- CSRs, *see* Control and Status Registers
- CSR SEL signal, 528, 529
- Customers Returns Area (CRA), 14
  
- data, floating point, formats for, 116-118
- Data Address Line Bus, 37
  - in LSI-11/23 microcomputer, 334-335
- Data Address Lines (DALs)
  - FPF11 floating-point processor interface with, 292, 296
  - in LSI-11/2 microcomputer, 37
  - in LSI-11/23 microcomputer, 334-335
- data chips
  - on LSI-11/2 microcomputer, 361
  - on LSI-11/23 microcomputer, 36-37, 331
- data-late errors, 344
- data read operations, on MRV11-AA memory, 415
- Data Systems, 10
- data transfers
  - bus cycles for, 223-233
  - direct memory access phase for, 234
  - in LSI-11/23 microcomputers, 340
  - in PDP-11/23-PLUS microcomputers, 618
- data word format (in PROMs), 466-467, 469, 470
- DATI bus cycle, 32
  - in LSI-11 Bus, 225-228
  - in MSV11-D, E memories, 502, 504
  - in MXV11-A memory and asynchronous serial line interface, 578
- DATIO bus cycle, 32, 344, 450, 461
- DATIO(B) bus cycle, 32, 230, 502, 578
- DATO(B) bus cycle, 32
  - in LSI-11 Bus, 229-230
  - in MSV11-D, E memories, 502, 505
  - in MXV11-A memory and asynchronous serial line interface, 578
- DATOB write cycle, 577
- DATO write cycle, 577
- DBIN L signal, 368
- DBOUT L signal, 368
- DC302 data chip, 331
- DC303 control chip, 331
- DCOK H signal, 345
- DDV11-B backplane, 39
- debugging
  - octal debugging technique for, 4, 161-182
  - ODT/ASCII console routine for, 4
- DECB instruction, 92
- DEC instruction, 92
- DECUS (Digital Equipment Computer Users Society), 13
- deferred addressing modes, 45
- destination field, 67

device addressing, 225  
 device priority, 237, 240-242  
 device registers, 556-557  
 device selection jumpers, 626-628  
 device vectors, 236  
 DF (displacement field), 280, 281  
 DIAGNOSE command, 473  
 diagnostic registers, 613-617  
 diagnostics  
   on FPF11 floating-point processor, 288-291  
   in PB11 universal PROM programmer, 470  
   in PDP-11/23-PLUS microcomputers, 610-612, 632-635  
   processor, 603  
 diagnostic switches and jumpers, 632-635  
 DIB (displacement in block), 280, 281  
 Digital Equipment Computers Users Society (DECUS), 13  
 DIN H signal, 373  
 DIP (Dual-In-Line Pack) switch units, 622, 628, 632  
 direct address mode, 45, 434-439, 459  
 direct address multiplexers, 459  
 direct memory access (DMA) in LSI-11 Bus, 234-236, 317  
   in LSI-11 microcomputer, 29, 31  
   in LSI-11/23 microcomputer, 330, 340-344  
 direct memory access grant (BDMG) signals, 220  
 disk bootstrap (MXV11), 564-565  
 displacement in block (DIB), 280, 281  
 displacement field (DF), 280, 281  
 display register, 615-616  
 distributed arbitration, 237  
 DIVD instruction, 131-132  
 DIVF instruction, 131-132  
 DIV instruction, 111  
 division, 124, 214-215  
 DLV11-KA converter option, 541, 566, 567, 569  
 DMA, *see* direct memory access  
 DMA arbitration logic, 373  
 DMA ENA flip-flop, 343  
 DMA latency, 344  
 DMGCY H signal, 361, 369, 373  
 documentation, for LSI-11 microcomputers, 10-11  
 \$ (dollar sign) ODT command, 167, 175  
 \$\$ (dollar sign, S) ODT command, 175  
 double-operand instructions, 65, 67-68, 73-74  
   format for, 45  
 double-precision floating point (double mode), 115, 116  
 double-precision integer long numbers, 118  
 DOUT L signal, 427  
 DRIVER ENABLE L inputs, 361  
 Dual-In-Line Pack (DIP) switch units, 622, 628, 632  
   early writes, 655  
 ED (expansion direction bit), 277-278  
 educational services, DIGITAL's, 11-12  
 EFCLR L signal, 370  
 EIA converter, 580  
 EIS (Extended Instruction Set), 4, 38, 77, 110-112  
 EMT instruction, 92-93, 211, 212  
 .ENABL AMA, 456

**EPROMs, 554, 634**  
 erasing, of PROMs, 467  
 errors  
   bus, 368-369  
   from clearing Interrupt Enable bits, 197-198  
   data-late, 344  
   in floating point, 122, 124  
   framing, 566  
   octal debugging technique and, 164, 165, 171  
   parity, 504-505, 520, 521, 658  
   PB11 universal PROM  
   programmer diagnostic messages for, 470  
   processor traps and, 210-211  
   time-out, 220, 461  
 event line interrupts, 370  
 event line (W4), jumper, 325  
 EVIRQ (1) H signal, 370  
 EVNT (1) H signal, 370  
 EVNT (external event) interrupts, 360, 385  
 exceptions, in floating point, 119, 122-123  
 exits, 70  
 expander boxes, 601  
 expansion direction (ED) bit, 277-278  
 expansion of systems, 601-603  
 exponents, 154  
 Extended Instruction Set (EIS), 4, 38, 77, 110-112  
 Extended LSI-11 Bus, 607  
 external event (EVNT) interrupts, 360, 385  
 EXT REF H signal, 496  
  
**FADD instruction, 153, 156**  
**Failed Systems Procedure, 392-393**  
  
**FALCON SBC-11/21**  
 microcomputers (KXT11-AA), 1-3, 299-318  
   architecture of, 36  
   maximum direct address space of, 32  
**FAR (First Address Range), 533, 535**  
**FD/FL register, 296**  
**FDIN (O) H signal, 366, 373**  
**FDIV instruction, 153, 156-157**  
**FEA (floating exception address register), 122-123, 125**  
**FEC (floating exception code register), 122-123, 125**  
**Field Programmable Logic Array (FPLA), 313**  
**First Address Range (FAR), 533, 535**  
**FIS (Floating Point Instruction Set), 3, 38, 153-158, 374, 386**  
**floating exception address register (FEA), 122-123, 125**  
**floating exception code register (FEC), 122-123, 125**  
**floating mode (single-precision floating point), 115, 116**  
**floating point data, 117-118**  
**Floating Point Instruction Sets FIS, 3, 38, 153-158, 374, 386**  
   FP-11, 115-150  
   FPF11 floating-point processor and, 287  
**Floating Point Option (KEF11), 4, 38, 153-156**  
   in PDP-11/23-PLUS microcomputers, 609  
**floating point option instruction addressing, 123**  
**floating-point processor (FPF11), 4, 38, 115, 116, 287-298**  
**floating-point register (FPIR), 292, 293, 296**

floating point status register (FPS), 118-122, 296-297  
 floating point zero, 116-117  
 FMUL instruction, 153, 157  
 formats  
   for condition code operators, 72-73  
   for double-operand instructions, 67-68  
   for FIS instruction set, 153  
   for floating point data, 116-118  
   for jump and subroutine instructions, 69, 70  
   for operand instructions, 44-45  
   for single-operand instructions, 66  
   for program control instructions, 69  
 FORTRAN (language), 38  
   for PROMs, 472  
 FORTRAN IV (language), 6  
 FORTRAN IV-PLUS (language), 7  
 FP-11 (KEF11-AA; Floating Point Instruction Set), 115-150  
   FPF11 floating-point processor and, 287  
   with LSI-11/23 microcomputer, 350-351  
 FP11A Floating Point Processor instruction set, 38  
 FPF11 floating-point processor, 4, 38, 115, 116, 287-298  
   with LSI-11/23 microcomputer, 351  
 FPIR (floating-point register), 292, 293, 296  
 FPLA (Field Programmable Logic Array), 313  
 FPS (floating point status register), 118-122, 296-297  
 fractions, 154  
 framing errors, 566  
 FSUB instruction, 153, 157-158  
 GADMR signal, 343  
 general-purpose registers (GPRs), 3  
   in FALCON SBC-11/21 microprocessor, 316  
   in LSI-11 microcomputer, 31  
   in LSI-11/2 microcomputer, 361  
   for memory reference addressing, 43-46  
   Program Counter as, 52-53  
   re-entrant code for, 202  
   saved by trap handlers, 211  
   used with FIS floating point, 155  
 G ODT command, 168, 175-176  
 go ODT command, 168, 175-176  
 GPRs, *see* general-purpose registers  
 H9270 card cage series, 39  
 H9275-A card cage, 39  
 H9276-A card cage, 39  
 H9281 card guide series, 39  
 half-duplex mode, 165  
 halt function, 243  
 HALT instruction, 39  
   on LSI-11 Bus, 244  
   in LSI-11/2 microcomputer, 366, 368  
   octal debugging technique and, 164  
   in PDP-11/23-PLUS microcomputer, 625, 626  
   in user mode, 273  
 HALT mode, 368  
   ODT Test Procedure for, 395  
 halt/reboot on break, 566  
 halt-trap option (W7), 329  
 hardware  
   FPF11 floating-point processor, 287-298  
   interrupts initiated by, 196

- for octal debugging technique, 179-182
- in OEM packages, 10
- PF11 universal PROM programmer option and, 461, 470-473
- required for troubleshooting, 389
- hardware stack pointer, 190
- HELP command, 473
- hidden bit, 116, 154
  
- IAK L signal, 369
- IDAL (Internal Data and Address Line) bus, 607
- IFETCH (instruction fetch), 292, 293
- immediate mode, 186
- inaccessible memory, 272
- INCB instruction, 94
- INC instruction, 94, 194-195
- index deferred mode, 52
- index mode, 51
  - position-independence and, 186
- index registers, 43
- indirect addressing modes, 45, 46
- industrial process control, 8-10
- INIT (1) H signal, 361, 369
- initialization
  - in LSI-11 Bus, 243
  - of LSI-11/23 clock generator circuitry, 345
- input buffers, 194
- installation, 14
  - of FPF11 floating-point processor, 288-289
  - of modules, 390-391
  - of MRV11-BA module, 420-421
  - of PROMs, 464, 467, 469
- instructions fetch (IFETCH), 292, 293
- instructions and instruction sets, 65-112
  - addressing modes and, 43, 44
  - Commercial Instruction Set, 584, 609
  - Extended Instruction Set, 4
  - FIS (floating point), 153-158
    - for floating point, 38
  - FP-11 (floating point), 115-150
    - for memory management, 284-285
  - PDP-11, 3, 35-36
  - reserved, 210
  - trap, 211-213
- integer data format, 118
- integration, levels of, 38-40, 583
- INTERFACE command, 473
- interface connector pins, 572
- interfaces
  - for FALCON SBC-11/21 microcomputer, 308, 316-317
  - between FPF11 floating-point processor and CPU, 292-262
  - LSI-11 Bus, 4, 29-31, 219-262
  - for MCV11-D memory, 537
  - MXV11-A, 541-580
  - for octal debugging technique, 179
- Internal Data and Address Line (IDAL), bus, 607
- internal register designer ODT command, 167, 175
- interrupt acknowledge flip-flop, 369
- interrupt acknowledge (BLACK; WIAK H) signals, 220, 369
- Interrupt Enable bits, 197-198
- interrupt exits, 70
- interrupt instructions, 70
- interrupt priority bit, 35
- interrupts, 196-200
  - in clock generator cycle, 349
  - in FALCON SBC-11/21 microcomputer, 313, 314



- in FIS floating point, 156
- in FP-11 floating point, 117, 122, 125
- in LSI-11 Bus, 236-242
- in LSI-11 microcomputer, 31
- in LSI-11/2 microcomputer, 366, 369-370
- LTC, 360, 385
- protection in, 275
- interrupt service routines, 196-197
- Interrupt Test Procedure, 399-401
- interrupt vectors, 561-562
- invalid ODT characters, 171
- I/O interrupts, normal, 369-370
- IOIRQ (1) H signal, 369
- I/O pages, 271, 535
- I/O timing
  - on MRV11-AA memory, 415
  - on MXV11-A memory and asynchronous serial line interface, 580
- IOT instruction, 94
- JAM MPC ZERO H signal, 295
- JMP instruction, 94-95
- JSR instruction, 69, 95-96
  - coroutine calls and, 203-204
  - stack use and, 191-193
  - subroutine linkage and, 195, 196
- jumper configurations
  - for LSI-11 microcomputers, 379-386
  - for LSI-11/2 microcomputers, 356-360
  - for LSI-11/23 microcomputers, 323-330
  - for MCV11-D memory, 530, 533, 535
  - for MRV11-AA memory, 406-410
  - for MRV11-BA memory, 417-419, 425
  - for MRV11-C memory, 437, 442, 444, 448, 459
  - for MSV11-B memory, 475, 476
  - for MSV11-CD memory, 481, 483-484
  - for MSV11-D, E memories, 489, 490, 496
  - for MSV11-L memory, 509, 511, 516, 520
  - for MSV11-P memory, 644-650
  - for MXV11-A memory and asynchronous line interface, 543-558, 562, 563, 565-567, 575, 578, 580
  - for PDP-11/23-PLUS microcomputers, 617, 622-632, 634-639
  - for PROMs, 461
  - see also* configurations
- jump instructions, 65, 69-70
- KDF11-AA, *see* LSI-11/23 microcomputers
- KDF11-AC microcomputers
  - contact finger identification of, 250
  - wake-up circuit on, 346
- KDF11-B, *see* PDP-11/23-PLUS microcomputers
- KEF11 (Floating Point Option), 4, 38, 153-156
- KEF11-A (Floating Point Option), 609
- KEF11-AA, *see* FP11
- kernel mode
  - interrupt and trap processing in, 275
  - memory management in, 265, 266, 271
- KEV11 (Floating Instruction Set; FIS), 4, 374, 386
- KXT11-AA, *see* FALCON SBC-11/21 microcomputers
- KXT11-AA TDAL bus, 316

- languages, 6-7
  - conversion routines for, 213
- LAT CSR SEL (L) signal, 529, 652
- LAT PAR ERR signal, 520
- layered products, 6
- LDCDF instruction, 132-133
- LDCFD instruction, 132-133
- LDCID instruction, 133-134
- LDCIF instruction, 133-134
- LDCLD instruction, 133-134
- LDCLF instruction, 133-134
- LDD instruction, 136
- LDEXP instruction, 134-136
- LDF instruction, 136
- LDFPS instruction, 136-137
- line clocks, 609
- line clock status register, 609-610
- line feed ODT command, 166-167, 173
- line time clock (LTC)
  - interrupts, 360, 370, 385
- Line Time Clock (LTC) Program, 398-399
- linkage of subroutines, 195-196
- linkage registers, 191-192
- linker (task builder), 185
- LIST command, 473
- loading
  - MRV11-C memory used for, 457-458
  - of PROMs, 462
- LOCKOUT signal, 538
  - in MSV11-D, E memories, 502
- L ODT command, 178-179
- LOW RANGE L signal, 496
- LSI-11 Bus (Q-Bus), 4, 29-31, 219-262
  - boards and, 39
  - on FALCON SBC-11/21
    - microcomputer, 310, 313, 314, 317
    - LSI-11 microcomputer used with, 377
    - LSI-11/2 microcomputer used with, 353
    - LSI-11/23 microcomputer used with, 340
    - MCV11-D memory interface with, 537
    - MRV11-C memory used with, 432
    - MSV11-D, E memories and, 496
    - PDP-11/23-PLUS backplane jumpers for, 635-639
    - peripheral devices compatible with, 330
    - width of addressing logic in, 266
- LSI-11 MDS development system, 40
  - LSI-11 microcomputers, 1, 337-386, 583
    - applications on, 8-10
    - architecture of, 29-35
    - documentation for, 10-11
    - educational services for, 12
    - features of, 3-4
    - FIS floating point on, 153-158
    - FP-11 floating point on, 115-150
    - instruction set for, 65-112
    - maintenance for, 13-17
    - memory reference addressing in, 43
    - octal debugging technique commands for, 171-179
    - parity functions not supported on, 505
    - stacks in, 189
    - troubleshooting on, 389-402
- LSI-11/2 microcomputers, 1-3, 353-375, 583
  - architecture of, 37-38
  - FIS floating point on, 153-158
  - maximum direct address space of, 32
  - octal debugging technique commands for, 171-179

LSI-11/23 (KDF11-AA)  
 microcomputers, 1-3, 321-351, 583  
   architecture of, 36-37  
   FP-11 floating point on, 115-150  
   FPF11 floating-point processor  
   with, 287  
   interrupt protocol on, 237-241  
   maximum direct address space  
   of, 32  
   octal debugging technique  
   commands for, 165-171  
   options with, 38  
 physical address space available  
 with, 265  
   width fo addressing logic in, 266  
 LTC (line time clock)  
 interrupts 360, 370, 385  
 LTC (Line Time Clock)  
 Program, 398-399  
 M7264, 385-386  
 M7265-YA, 385-386  
 MACRO-11 (language), 6  
   deferred addressing modes used  
   with, 45  
   output of, 185  
   recursion within, 209  
 maintenance, 13-17  
 maintenance ODT command, 176-  
 177  
 manufacturing test jumpers, 624  
 mapping, 265  
 MARK instruction, 96  
 master clock (WI) jumper, 324  
 master colcks, 336  
 master-slave  
 communications, 29, 220  
 MATCH 256 H signal, 425, 428  
 MATCH 4K H signal, 426  
 MCLK H signal, 345  
 M command, 395  
 MCV11-DA memory, 530  
 MCV11-DC memory, 530  
 MCV11-D memory, 530-538  
 memories  
   addressing of, 32-33, 43  
   addressing of, by FALCON SBC-  
   11/21  
     microcomputer, 310-313  
   addressing of, by LSI-11 Bus,  
   225  
   failed systems procedure and,  
   392  
   in FALCON SBC-11/21  
   mocrocomputer, 317-318  
   in LSI-11/23 microcomputer, 330  
 MCV11-D, 530-538  
 MRV11-AA, 405-415  
 MRV11-BA, 416-431  
 MRV11-C, 432-459  
 MSV11-B, 475-478  
 MSV11-CD, 479-484  
 MSV11-D and MSV11-E, 485-505  
 MSV11-L, 506-529  
 MSV11-P, 641-662  
 MXV11-A, 541-580  
 PROMs, 461-473  
   protection of, 271-275  
   stacks and, 189  
 memory arrays  
   in MCV11-D memory, 538  
   in MSV11-D, E memories, 494-  
   496  
   in MSV11-L memory, 519  
   in MSV11-P memory, 655-656  
   in MCV11-A memory and  
   asynchronous serial line  
   interface, 574-575  
 memory bank decoder, 578  
 memory management, 265-285  
   FP-11 floating point and, 115,  
   116  
 memory management cycle, 348-  
 349  
 Memory Management Unit  
 (MMU), 38, 265, 268

- on LSI-11/23 microcomputer, 331, microcomputers 351
- in PDP-11/23-PLUS processor, , 607
- required for KEF11-AA option, , 116
- memory modules, 330
- memory refresh
  - in LSI-11 Bus, 243
  - in LSI-11 microcomputer, 385-386
  - in MSV11-B memory, 475, 476, 478
  - in MSV11-CD memory, 483
  - in MSV11-D, E memories, 502, 503
  - in MSV11-L memory, 517, 523-525
  - in MSV11-P memory, 660
  - MCV11-A memory and asynchronous serial line interface, 577
- memory relocation, 267-271
- Memory Resident RT (MMRT; operating system), 10
- metal oxide semiconductor memories, *see* MOS memories
- MFPD instruction, 97, 285
- MFPI instruction, 97, 285
- MFPS instruction, 97
- MFPT instruction, 98
- MIB, *see* Microinstruction Bus
- MIB03/GPO 3, 377
- MIB12, 9, 8/Address-Input-Output (AIO) codes, 335-336
- MIB 13/Interrupt Acknowledge (IAK), 335
- MIB 14/Initialize (INIT F), 335
- MIB 15/Memory MANagement Enable (mme), 335
- microcoded floating point, 4
- microdode ODT, 161-182
- microcomputer boxes, 40
- FALCON SBC-11/21, 299-318
- LSI-11, 377-386
- LSI-11/2, 353-375
- LSI-11/23, 321-351
- PDP-11/23-PLUS, 605-639
- microcontroller (in FPF11), 293-295
- Micronstruction Bus (MIB)
  - FPF11 floating-point processor interface with, 292, 295
  - on LSI-11 1/2 microcomputer, 37, 360
  - on LSI-11/23 microcomputer, 37, 335-337
  - on PDP-11/23-PLUS microcomputer, 607
- MICROM chips, 37
- MicroPower/Pascal (operating system), 5, 7, 299
- microprocessor chip sets, *see* chip sets
- microprocessor data path logic, 297
- microprocessors
  - in FALCON SBC-11/21 microcomputer, 314, 316
  - see also* processors
- microprocessor unit (MPU), 36
- miscellaneous instructions, 65, 70-71
- MMU, *see* Management Unit
- MMU chip, 351
- MODD instruction, 137-140
- mode register, 316
- MODF instruction, 137-140
- MODIFY command, 473
- M ODT command, 176-177
- modules
  - contact finger identification of, 250-252
  - installation of, 390-391
  - MRV11-C memory, 432

- relocatable object, 185
  - see *also* memories
- module selection, 537
- module selection decode, 516
- Module Starting Address (MSA)
  - in MCV11-D memory, 533-535
  - in MSV11-L memory, 509
  - in MSV11-P memory, 644-645
- MOS (metal oxide semiconductor) memories
  - MSV11-CD, 479-484
  - MSV11-D, E, 485
  - MSV11-L, 519
  - MSV11-P, 641-662
- MOS memory address logic, 516-518, 650-652
- mounting boxes, 39-40
- MOVB instruction, 98-99, 194-195
- MOV instruction, 98-99, 462
- MPU (microprocessor unit), 36
- MRV11-AA memory, 405-415
  - PROMs for, 451-464
- MRV11-BA memory, 416-431
  - PROMs for, 461, 466-467
- MRV11-BC memory, 416, 420, 421
- MRV11-C memory, 432-459
  - PROMs for, 461, 467-469
- MSA, see Module Starting Address
- MSEL L signal, 537
- MSV11-B memory, 475-478
- MSV11-CD memory, 479-484
- MSV11-DA memory, 485, 490, 494
- MSV11-DB memory, 485, 490, 494
- MSV11-DC memory, 485, 490, 494
- MSV11-DD memory, 485, 490, 494
- MSV11-D memories, 485-505
- MSV11-EA memory, 485, 490, 494
- MSV11-EB memory, 485, 490, 495
- MSV11-EC memory, 485, 490, 495
- MSV11-ED memory, 485, 490, 495
- MSV11-E memories, 485-505
- MSV11-LF memory, 506
- MSV11-LK memory, 506
- MSV11-L memory, 506-529
- MSV11-PK memory, 641
- MSV11-PL memory, 641
- MSV11-P memory, 584, 641-662
- MTPD instruction, 99, 285
- MTPI instruction, 99, 285
- MTPS instruction, 100, 273-275
- MULD instruction, 140-141
- MULF instruction, 140-141
- MUL instruction, 112
- multiple address space, 266
- multiplexers, 459
- multiplication, 215-217
- multiprogramming environments, 185, 201, 265
  - protection in, 271-275
- multitask program environments, 201
- multi-user BASIC (language), 7
- MXV11-AA memory, 469-470, 574
- MXV11-AC memory, 469-470, 574-575
- MXV11-A memory and asynchronous serial line interface, 541-580
  - PROMs for, 461, 469-470
- MXV11 disk bootstrap, 564-565
- MXV11 TU58 bootstrap, 563-564
- N bit, 71
- NEGB instruction, 100
- NEGD instruction, 141-142
- NEGF instruction, 141-142

NEG instruction, 100  
 nesting  
   of interrupts, 198-200  
   of subroutine calls, 196  
 nonvanishing floating point numbers, 116  
 No Operation (NOP) Program, 395-398  
 normal (clock) cycle, 347  
 no SACK timeout signal, 342, 343  
 notation, in instruction set, 75-76  
  
 octal debugging technique (ODT), 4, 161-182  
 odd addressing errors, 210  
 ODT-11, 161  
 ODT/ASCII console routine, 4, 37  
 ODT PROMPT, 389, 391  
 ODT Test Procedure, 395  
 OEM hardware/software packages, 10  
 offsite service, 14-17  
 on-site service, 13-14  
 op code, 65  
 operand instructions, 44-45  
 operating systems  
   MicroPower/Pascal, 299  
   MRRT, 10  
   RSM-11M, 2, 5-6  
   RSTS/E, 6  
   RT-11, 5  
   runtime application support for, 7-8  
 options  
   DLVaa-DA converter, 541, 566, 567, 569  
   with FALCON SBC-11/21 microcomputers, 36  
   FPF11 floating-point processor, 287  
   with LSI-11 microcomputers, 386  
   with LSI-11/2 microcomputers, 38, 374  
   with LSI-11/23 microcomputers, 330, 350-351  
   PB11 universal PROM programmer, 461, 470-473  
   with PDP-11/23-PLUS microcomputers, 609  
 overflows, floating, 117, 124, 125  
  
 packaged development systems, 40  
 page address field (PAF), 276, 280, 281  
 page address register (PAR), 266, 272, 276, 281  
 page control register (PCR), 614  
 page descriptor register (PDR), 266, 272, 276-279, 281  
 page length field (PLF), 279  
 pages, 269-271  
 PALs (Programmable Logic Arrays), 331, 529  
 PAR, *see* page address register  
 parallel I/O interfaces, 308, 317  
 parallel port addresses, 313, 314  
 parameter jumpers, 562  
 PAR ERROR IND H signal, 504  
 parity checkers, 658-659  
 parity control circuitry, 656  
 parity generation, 658  
 parity logic  
   in MSV11-E memories, 504-505  
   in MSV11-L memory, 519-521  
   in MSV11-P memory, 658-659  
 Partial Starting Address (PSA), 533, 535  
 PASCAL (language), 7  
 patch areas, 211

**PB11-AD universal PROM programmer, 472**  
**PB11-AQ universal PROM programmer, 472**  
**PD11-AY universal PROM programmer, 471**  
**PB11K-AA universal PROM programmer, 472**  
**PB11K-AB universal PROM programmer, 472**  
**PB11K-AC universal PROM programmer, 472**  
**PB11 universal PROM programmer, 40, 461, 470-473**  
**PC, see Program Counter**  
**PC immediate mode, 53-54**  
**PCR (page control register), 614**  
**PC relative deferred mode, 55-56**  
**PC relative mode, 54-55**  
**PDP-11 instruction set, 3, 35-36**  
**PDP-11 systems, 1, 583**  
     features of, 3-4  
     LSI-11/23 microcomputer compatibility with, 321  
     maintenance for, 13-17  
     memory management compatibility with, 265  
     memory reference addressing in, 43  
     position-independent code in, 185  
     stacks in, 198  
     word length in, 266  
**PDP-11/03 systems, 1, 3, 583, 587, 595-600**  
     FIS floating point on, 153-158  
**PDP-11/03-L systems, 583, 586, 602-603**  
**PDP-11/23 systems, 1, 3, 583**  
     expansion of, 602  
     FP-11 floating point on, 115-150  
     FPF11 floating-point processor with, 287  
     functionality and performance of, 584  
     in packaged systems, 595-600  
     peripheral options compatible with, 380  
     processor self-diagnosis in, 603  
**PDP-11/23-BC processor, 584**  
**PDP-11/23-BE processor, 584**  
**PDP-11/23-PLUS microcomputers, 1-3, 583, 605-639**  
     expansion of, 601-602  
     FP-11 floating point on, 115-150  
     FPF11 floating-point processor with, 287  
     functionality and performance of, 584  
     in packaged systems, 589-590  
     processor self-diagnosis in, 603  
**PDP-11T03-L systems, 595, 601**  
**PDP-11T23 systems, 595, 601**  
**PDP-11V03-L systems, 597, 601**  
**PDP-11V23 systems, 597, 601**  
**PDR, see page descriptor register**  
**peripherals, 5**  
     addressing of, 32-33  
     addressing of, by LSI-11 Bus, 225  
     four-level interrupt configurations for, 241-242  
     interrupts initiated by, 196  
     with LSI-11/23 microcomputers, 330  
**PFAIL H signal, 366, 367**  
**physical addresses (PA), 267-269, 279-281**  
**physical address space, 265**  
**pin identification system, 250**  
**PLAs (programmable logic arrays), 331, 529**  
**PLF (page length field), 279**  
**P ODT command, 169, 176**  
**pointers, 43**  
**popping (of stack data), 190-191**

- position-defined arbitration, 237
- position-dependent code, 185, 187-189
- position-independent code, 185, 189
- post-warranty service, 15
- power controllers, 602-603
- power-fail/auto restart logic, 4
- power-fail/restart sequence, 366-368
- power failures, processor traps and, 210
- power jumpers, 649, 660
- power status protocol, 243
- power supply, for LSI-11 Bus, 250
- power-up, on FALCON SBC-11/21 microcomputer, 307
- power-up/down protocol, 343-344
- power-up mode selection
  - in LSI-11 microcomputers, 383-385
  - in LSI-11/2 microcomputers, 356-360
  - in LSI-11/23 microcomputers, 325-329
  - in PDP-11/23-PLUS microcomputers, 624-625
- PPI (Programmable Peripheral Interface), 308
- precision, in floating point, 123-125
- priority levels
  - device, 237, 240-242, 313, 314
  - interrupt priority bit for, 35
  - of interrupts and DMA, 31
  - in LSI-11/2 microcomputers, 369
  - in LSI-11/23 microcomputers, 334
- proceed ODT command, 169, 176
- processor I/O addresses, ODT and, 170
- processors (central processing units; CPUs), 584-586
  - bus arbitration by, 220
  - bus cycles in, 32
  - floating point on, 115
  - FPF11 floating-point option for, 287
  - general-purpose registers in, 31
  - interface between FPF11 floating-point processor and, 292-293
  - interrupts in, 196
  - LSI-11 Bus signals to, 29
  - MPU, 36
  - octal debugging technique and, 161
  - PSP-11/23-PLUS, 607-609
  - self-diagnosis on, 603
  - stacks in, 189
  - standard features of, 3-4
  - traps and, 210-211
  - used in troubleshooting, 389
- Processor Status Word (PS; PSW), 33, 34
  - in FALCON SBC-11/21 microprocessor, 316
  - interrupt priority bit and, 35
  - in interrupt, 197
  - in interrupt and trap processing, 275
  - LSI-11 Bus interrupts and, 236
  - in LSI-11/23 microcomputers, 332-334, 340
  - mode specification in, 272-273
  - trap instructions and, 211
- processor status word ODT command, 168, 175
- processor traps, 210-211
- Product Repair Centers (PRCs), 14
- program control instructions, 68-69
- Program Counter (PC; R7), 3, 31, 43, 52-53
  - in FALCON SBC-11/21 microprocessor, 316
  - in interrupts, 197
  - LSI-11 Bus interrupts and, 237
  - stack use and, 192
  - trap instructions and, 211



program counter addressing modes, 46, 52-56

Programmable Logic Arrays (PALs), 331, 529

Programmable Peripheral Interface (PPI), 308

programmable read-only memories, see PROMs

programming, 185-217

- with MRV11-C memory, 455-458
- PB11 universal PROM programmer for, 470-473
- of PROMs, 461-462

programs

- protection for, 271-275
- relocation of, 269-271

PROM address jumpers, 419

PROMs (programmable read-only memories), 461-473

- MCV11-D memory and, 537
- MRV11-AA, 405-415
- MRV11-BA, 419, 421-422
- MSV11-L memory and, 516
- in MSV11-P memory, 649-652
- in MXV11-A memory and asynchronous serial line interface, 554, 575
- non-volatility of contents of, 416
- PB11 PROM Programmer for, 40

protection of memory, 271-275

PS, see Processor Status Word

PSA (Partial Starting Address), 533, 535

PSW, see Processor Status Word

pure code, 201

pushing (of stack data), 190

Q-Bus, see LSI-11 Bus

QJV11 ROM/PROM formatter software, 416, 461

radix points, 154

RAM, see random-access memories

RAM address jumpers, 419

RAM read sequene, 428-429

RAM SEL L signal, 575

RAM write sequence, 429

random-access memories (RAM)

- on FALCON SBC-11/21 microcomputer, 317-318
- MCV11-D, 530-538
- MRV11-BA, 416-431
- MSV11-B, 475-478
- MSV11-CD, 479-484
- MSV11-D, and MSV11-E, 485-505
- MSV11-L, 506-529
- MSV11-P, 641-662
- on MXV11-A memory and asynchronous serial line interface, 543, 554, 574-578

RAS (row address strobes), 518, 525, 575

RASO (row address strobe O), 518, 519

RAS SEL H signal, 518

RAS SEL (L) signal, 518, 519

RBUF (receiver buffer register), 164, 180-181, 618-620

RCSR (receiver control and status register), 180, 559, 618-619

read-only memories (ROM), 272, 331

- on FALCON SBC-11/21 microcomputer, 318
- MRV11-AA, 405-415
- MRV11-BA, 416-431
- MRV11-C, 432-459
- on MXV11-A memory and asynchronous serial line interface, 543, 554-555, 578-580
- PROMs for, 461-473

read/write maintenance register, 615

real-time clock input, 4  
 receiver buffer register (RBUF), 164, 180-181, 618-620  
 receiver control and status register (RCSR), 180, 559, 618-619  
 receiver data register, 560  
 REC H signal, 249  
 recursion, 208-209  
 re-entrancy, 200-203  
 re-entrant code, 201-203  
 REF ADD L signal, 577  
 REF REQ (L) signal, 523  
 refresh, *see* memory refresh  
 refresh jumpers, 476  
 register deferred mode, 47-48  
 register mode, 46-47  
 registers, 3  
   in FALCON SBC-11/21 microprocessor, 316  
   for FP-11 floating point processor, 118-123  
   for FIS floating point, 154-155  
   general-purpose, 31  
   in LSI-11/2 microcomputer, 361  
   for memory management, 266, 268, 269, 276-279  
   for memory mapping, 265  
   for memory reference addressing, 43-46  
   in MSV11-L memory, 526-529  
   in MXV11-A memory and asynchronous serial line interface, 556-561  
   for octal debugging technique, 180-182  
   in PDP-11/23-PLUS microcomputers, 609-610, 612-622  
   status, 282-284  
   TBus, 296-297  
 relative addressing modes, 186  
 relocatable object modules, 185  
 REPLY (1) H signal, 369  
 reply jumpers, 406-410  
 reserved instructions, 210  
 reserved ODT commands, 169  
 reset cycle, 349  
 RESET instruction, 101, 273  
   in FPF11 floating-point processor, 295  
   in LSI-11 Bus, 243  
   in LSI-11/23 microcomputer, 349  
   in MRV11-C memory, 447  
 reset logic, 366  
 RESET L signal, 367-369  
 REV11 refresh/booster module, 478  
 R ODT command, 167, 175  
 ROLB instruction, 101-102  
 ROL instruction, 101-102  
 ROM, *see* read-only memories  
 ROM chips, 447-450  
 ROM codes, 370-371  
 ROM L signal, 578  
 RO (rubout) ODT command, 177-178  
 RORB instruction, 102  
 ROR instruction, 102  
 rounding, 118, 124  
 row address strobe O (RASO), 518, 519  
 row address strobes (RAS), 518, 525, 575  
 row address strobe signals, 495  
 RPLY L signal, 577  
 RS232-C PROM blaster, 461  
 RSTS/E (Resource Sharing Timesharing System/Extended; operating system), 6  
 RSX-11M (operating system), 2, 5-6

languages supported under, 6-7  
 RSX-11S (operating system), 7-8  
 RT2 (operating system), 8  
 RT-11 (operating system), 5  
   languages supported under, 6-7  
   MRRT subset of, 10  
   PROMs and, 461, 470, 472  
 RT-11 bootstrap, 463  
 RT11 FORTRAN IV V2.5  
 (language), 472  
 RTI instruction, 102, 197, 198  
 RTS instruction, 70, 103, 196, 198  
 RTT instruction, 103  
 rubout (RO) ODT command, 177-178  
 run-time application support, 7-8

**SBC-11/21, see FALCON SBC-11/21 microcomputers**  
 SBCB instruction, 104  
 SBC instruction, 104  
 SCC instruction, 104  
 SEC instruction, 104  
 SEL2 signal  
   in MRV11-BA memory, 427, 428  
 SELECT L signal, 496, 502  
 self-diagnosis, on processors, 603  
 SEL RAS O signal, 518  
 SEN instruction, 105  
 sequencer (in FPF11), 293-295  
 SEQUENTIAL command, 477  
 serial line addresses, 313, 314  
 serial line interface (MXV11-A), 580  
 serial line parameter jumpers, 562  
 serial line registers, 556-557  
 serial line unit registers, 618-622

serial line units (SLUs), 580  
   in FALCON SBC-11/21 microcomputer, 316-317  
   in PDP-11/23-PLUS microcomputer, 617-618, 628-632  
 service, for LSI-11 based systems, 13-14  
 service agreements, 13-14  
 service routines, 236  
 SETD instruction, 142-143  
 SETF instruction, 142  
 SETI instruction, 143  
 SETL instruction, 143  
 SEV instruction, 105  
 SEZ instruction, 105  
 SFR register, 297  
 signal lines, in LSI-11 Bus, 219, 220  
 signed conditional branch instructions, 68  
 SIMRT (operating system), 8  
 single-operand instructions, 65, 66, 73  
   formats for, 44-45  
 single-precision floating point (floating mode), 115, 116  
 single-precision integers, 118  
 single-step circuits, 346  
 / (slash) ODT command, 165-166, 172  
 slave devices, 29, 220, 224, 225  
 SLUs, *see* serial line units  
 SOB instruction, 106  
 S ODT command, 168  
 software  
   executed in windowed programs, 455  
   in kernel mode, 265  
   in OEM packages, 10

PB11 universal PROM programmer option and, 470-473  
 for PROMs, 461  
 QJV11 ROM/PROM formatter, 416  
 software development tools, 5-8  
   LSI-11 MDS development system, 40  
 software exits, 70  
 Software Services, 14  
 source field, 67  
 SP, *see* Stack Pointer  
 special control function, 370-371  
 special purpose registers, 3  
 special symbols, in instruction set, 75-76  
 SR0M H signals, 370  
 SSPARE 3 signal, 459  
 stack addressing, 43  
 Stack Pointer (SP; R6), 3, 31, 43, 189-190, 192-193, 195  
   in FALCON SBC-11/21 microprocessor, 316  
   octal debugging technique and, 170-171  
   Processor Status Word and, 273  
   re-entrant code and, 202  
   used with FIS floating point, 155  
 stacks, 189-195  
   coroutines and, 203-205  
   recursion and, 208  
   re-entrancy for, 200-201  
 starting addresses  
   on FALCON SBC-11/21 microcomputer, 307  
   on LSI-11/23 microcomputer, 329-330  
   on MRV11-C memory, 437-439, 443-444  
   *see also* Module Starting Address  
 status register 0 (SR0), 282-283  
   status register 1 (SR1), 283  
   status register 2 (SR2), 284  
   status register 4 (SR3), 284  
   status registers, 282-284  
   STCDF instruction, 143-144  
   STCDI instruction, 145-146  
   STCDL instruction, 145-146  
   STCFD instruction, 143-144  
   STCFI instruction, 145-146  
   STCFL instruction, 145-146  
   STD instruction, 145  
   STEXP instructions, 147  
   STF instruction, 145  
   STFPS instruction, 147  
   STST instruction, 123, 148  
   SUBD instruction, 148-150  
   SUBF instruction, 148-150  
   SUB instruction, 106  
   subroutine calls, 455-456  
   subroutine instructions, 65, 69-70  
   subroutines  
     coroutines and, 203, 205  
     linkage of, 195-196  
   re-entrancy in, 201  
   stack use in calling of, 191-195  
   SWAB instruction, 107  
   switch configurations, 622, 628, 630-634  
   SXT instruction, 107  
   SYDMR signal, 343  
   SYNCF signal, 339  
   SYNCH/DMA ENA H signal, 336-337  
   SYNC H signal, 502  
   SYNC L signal, 575  
   System Check-Put Procedure, 400, 401  
   systems, 40, 583-603

system size jumpers, 649  
 task builder (linker), 185  
 tasks, 185  
     coroutines used with, 206  
     handling of several, 201  
     re-entrant routines requested by, 202  
 T bit (trap), 35  
 TBUF (transmitter buffer register), 618, 622  
 TBus, 296-297  
 TCSR (transmitter/control status register), 618, 621  
 TDAL bus, 316  
 Technical packaged systems, 589  
 terminals, Check-Out Procedure for, 391  
 TERR (1) H signal, 369  
 TERR L signal, 369  
 TFCLR L signal, 369  
 timing and control logic  
     in MCV11-D memory, 538  
     in MSV11-D, E memories, 498-503  
     in MSV11-L memory, 522-523  
     in MSV11-P memory, 656  
     in MXV11-A memory and asynchronous serial line interface, 575  
 time-out errors, 210, 220, 461  
 T PAR ERR L signal, 658  
 trace bits, 333  
 transmitter buffer register (TBUF), 618, 622  
 transmitter buffer register (XBUF), 163, 181-182  
 transmitter/control status register (TCSR), 618, 621  
 transmitter control and status register (XCSR), 181  
 transmitter data register, 561  
 transmitter status register, 560  
 trap T bit), 35  
 trap exits, 70  
 trap handlers, 211-212  
 TRAP instruction, 108, 211, 212  
 trap instructions, 65, 70, 211-213  
 traps, 211-213  
     for FIS floating point, 155  
     LSI-11 Bus and, 237  
     priority for, 369  
     processor, 210-211  
     protection in processing of, 275  
 trap vectors, 211, 213, 275  
 troubleshooting, 387-402  
 TRPLY signal, 655  
 TSTB instruction, 108  
 TSTD instruction, 150  
 TSTF instruction, 150  
 TST instruction, 108  
 UART (universal asynchronous receiver/transmitter) chips, 580, 629, 630  
 ultraviolet (UV) memories, 416  
     PROMs, 466  
 undefined variables, 117  
 underflows, floating, 117, 124, 125  
 UNIBUS, time-out errors and, 210  
 universal asynchronous receiver/transmitter (UART) chips, 580, 629, 630  
 unsigned conditional branch instructions, 68  
 ODT  
     command, 173-174  
     user mode  
         memory management in, 265, 266  
         restrictions in, 273-275

- V bit, 72
- vector addresses, 197, 211, 213
- vectors, interrupt, 561-562
- VERIFY command, 473
- virtual addresses (VA), 267-269, 279-281
- virtual address space, 268-271
  - in MRV11-C memory, 434
- voice data-entry systems, 10
- volatility of memory, 416, 479, 485
  
- WAIT instruction, 109
- wake-up circuits, 346, 372-373
- warranty service, 14-15
- W (written into) bits, 278
- WDAL bus, 360, 361
- WDIN H signal, 369
- WIAK H signal, 369
- window mapped mode, 434, 440-444, 447, 455-459
- word addressing, 33, 65
- word format (in PROMs), 463-464, 466-467, 470
- words, 266
  - division of, 32
- written into (W) bit, 278
- WTBT H signal, 502
- WTBT L signal, 495, 502
- WTBT0 L signal, 495
- TBTY1 L signal, 495
- WTBT signal, 537-538
- WTBYTE signal, 538
- WTHB L signal, 575, 577
- WTLB L signal, 575, 577
  
- XBUF (transfer buffer register), 163, 181-182
- XCSR (transmitter control and status register), 181
- XOR instruction, 109
- X-ray analysis, 10
  
- Z bit, 71
- zero, 116-117
  - in operands, 126









Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our handbooks.

What is your general reaction to this handbook? (format, accuracy, completeness, organization, etc.) \_\_\_\_\_

---

---

---

What features are most useful? \_\_\_\_\_

---

---

---

Does the publication satisfy your needs? \_\_\_\_\_

---

---

What errors have you found? \_\_\_\_\_

---

---

---

Additional comments \_\_\_\_\_

---

---

Name \_\_\_\_\_

Title \_\_\_\_\_

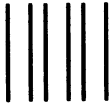
Company \_\_\_\_\_ Dept. \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

(staple here)

----- (please fold here) -----



**No Postage  
Necessary  
if Mailed in the  
United States**

**BUSINESS REPLY CARD**  
FIRST CLASS PERMIT NO. 33 MAYNARD, MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

**DIGITAL EQUIPMENT CORPORATION  
NEW PRODUCTS MARKETING  
PK3-1/M92  
MAYNARD, MASS. 01754**



**digital**

**HANDBOOK SERIES**

**Microcomputers and Memories**

**Microcomputer Interfaces**

**PDP-11 Processor**

**PDP-11 Software**

**Peripherals**

**Terminals and Communications**

**VAX Architecture**

**VAX Software**

**VAX Hardware**

# digital

DIGITAL EQUIPMENT CORPORATION, Corporate Headquarters: Maynard, MA 01754 Tel. (617) 897-5111 — SALES AND SERVICE OFFICES; UNITED STATES — ALABAMA, Birmingham, Huntsville ARIZONA, Phoenix, Tucson ARKANSAS, Little Rock CALIFORNIA, Bakersfield, Costa Mesa, El Segundo, Fresno, Los Angeles, Oakland, Sacramento, San Diego, San Francisco, Monrovia, Pasadena, Santa Barbara, Santa Clara, Santa Monica, Sherman Oaks, Sunnyvale COLORADO, Colorado Springs, Denver CONNECTICUT, Fairfield, Meriden DELAWARE, Newark, Wilmington FLORIDA, Jacksonville, Melbourne, Miami, Orlando, Pensacola, Tampa GEORGIA, Atlanta HAWAII, Honolulu IDAHO, Boise ILLINOIS, Chicago, Peoria INDIANA, Indianapolis IOWA, Bettendorf KENTUCKY, Louisville LOUISIANA, Baton Rouge, New Orleans MAINE, Portland MARYLAND, Baltimore, Odenton MASSACHUSETTS, Boston, Burlington, Springfield, Waltham MICHIGAN, Detroit, Kalamazoo MINNESOTA, Minneapolis MISSOURI, Kansas City, St Louis NEBRASKA, Omaha NEVADA, Las Vegas, Reno NEW HAMPSHIRE, Manchester NEW JERSEY, Cherry Hill, Parsippany, Princeton, Somerset NEW MEXICO, Albuquerque, Los Alamos NEW YORK, Albany, Buffalo, Long Island, New York City, Rochester, Syracuse, Westchester NORTH CAROLINA, Chapel Hill, Charlotte OHIO, Cincinnati, Cleveland, Columbus, Dayton OKLAHOMA, Tulsa OREGON, Eugene, Portland PENNSYLVANIA, Allentown, Harrisburg, Philadelphia, Pittsburgh RHODE ISLAND, Providence SOUTH CAROLINA, Columbia, Greenville TENNESSEE, Knoxville, Memphis, Nashville TEXAS, Austin, Dallas, El Paso, Houston, San Antonio UTAH, Salt Lake City VERMONT, Burlington VIRGINIA, Arlington, Lynchburg, Norfolk, Richmond WASHINGTON, Seattle, Spokane WASHINGTON D.C. WEST VIRGINIA, Charleston WISCONSIN, Madison, Milwaukee INTERNATIONAL — EUROPEAN AREA HEADQUARTERS: Geneva, Tel: [41] (22)-93-33-11 INTERNATIONAL AREA HEADQUARTERS: Acton, MA 01754, U.S.A., Tel (617) 263-6000 ARGENTINA, Buenos Aires AUSTRALIA, Adelaide, Brisbane, Canberra, Darwin, Hobart, Melbourne, Newcastle, Perth, Sydney, Townsville AUSTRIA, Vienna BELGIUM, Brussels BRAZIL, Rio de Janeiro, Sao Paulo CANADA, Calgary, Edmonton, Hamilton, Halifax, Kingston, London, Montreal, Ottawa, Quebec City, Regina, Toronto, Vancouver, Victoria, Winnipeg CHILE, Santiago DENMARK, Copenhagen EGYPT, Cairo ENGLAND, Basingstoke, Birmingham, Bristol, Ealing, Epsom, Leeds, Leicester, London, Manchester, Newmarket, Reading, Welwyn FINLAND, Helsinki FRANCE, Bordeaux, Lille, Lyon, Marseille, Paris, Puteaux, Strasbourg HONG KONG INDIA, Bangalore, Bombay, Calcutta, Hyderabad, New Delhi IRELAND, Dublin ISREAL, Tel Aviv ITALY, Milan, Rome, Turin JAPAN, Fukuoka, Nagoya, Osaka, Tokyo, Yokohama KOREA, Seoul KUWAIT, Safat MEXICO, Mexico City, Monterrey NETHERLANDS, Amsterdam, The Hague, Utrecht NEW ZEALAND, Auckland, Christchurch, Wellington NIGERIA, Lagos NORTHERN IRELAND, Belfast NORWAY, Oslo PERU, Lima PEURTO RICO, San Juan SAUDI ARABIA, Jeddah SCOTLAND, Edinburgh, REPUBLIC OF SINGAPORE, SPAIN, Barcelona, Madrid SWEDEN, Gothenburg, Stockholm SWITZERLAND, Geneva Zurich TAIWAN, Taipei TRINIDAD, Port of Spain VENEZUELA, Caracas WEST GERMANY, Berlin, Cologne, Frankfurt, Hamburg, Hannover, Munich, Nuremberg, Stuttgart YUGOSLAVIA, Belgrade, Ljubijana, Zagreb

ORDER CODE: EB-20912-20