EK-1184E-TM-001

PDP-11/84 System

Technical and

Reference Manual

digital™

EK-1184E-TM-001

PDP-11/84 System

Technical and

Reference Manual

Prepared by Educational Services
of Digital Equipment Corporation

December 1987

DEC	Micro/RSX	RSTS
DECmate	MicroVAX	RSX
DECnet	PDP	RT
DECUS	PDP-11	ULTRIX
DECwriter	PDP-11/84	UNIBUS
DIBOL	P/OS	VAX
J-11	Professional	VAXstation
MASSBUS	Q-bus	VMS
Micro/PDP-11	Q22-bus	VT
Micro/RSTS	Rainbow	Work Processor

digital™

This document was prepared using VAX DOCUMENT, Version 1.0

Contents

Examples

About This Manual

This manual provides a detailed functional description of the standard PDP-11/84 system.

The manual consists of the following chapters and appendices.

Chapter 1, System Overview - Includes a general description of the PDP-11/84 system and its components and a list of related documents.

Chapter 2, Functional Description - Describes each of the major logic elements of the PDP-11/84 system.

Chapter 3, Bootstrap and Diagnostic ROM Programming - Describes the commands for the ROM and EEPROM codes.

Appendix A, CPU Instruction Timing - Describes the method of calculating the execution time for CPU instructions.

Appendix B, Hardware/Software Differences - Describes the differences between the UNIBUS power-up protocol for the PDP-11/84 and other PDP-11 systems.

Appendix C, Configuration Register Modification - Describes the format of the boot and diagnostic configuration register (BCR).

Appendix D, Floating-Point Instruction Timing - Contains information necessary to calculate the floating-point instruction timing.

Appendix E, ROM Code Differences - Explains the differences between versions 6.0, 7.0, and 8.0 of the ROM code used in the KDJ11-BF CPU module.

Appendix F, Multiboot ROM Control Transfer - Describes the program used to transfer control to any UBA ROM or M9312 ROM boot.

Conventions Used in This Manual

Warnings, cautions, and notes appear throughout this guide. They serve the following functions.

Warning	Provides information to prevent personal injury.
Caution	Provides information to prevent damage to equipment or software.
Note	Provides general information about the current topic.

Other conventions used in this manual are as follows.

- Bits are usually shown in angle brackets. Two numbers separated by a colon indicate a set of bits, or bit field. For example, <15:08> stands for bits 15 through 8.
- Keyboard function keys are enclosed in a box. For example, the carriage return key is specified as $\boxed{\text{Return}}$.
- In the examples of screen display, the text the user enters is in bold type.

Some functions may be labeled differently on different keyboards. See the user's manual for your terminal if you have a question.

1
System Overview

1.1 Introduction

The PDP-11/84 system is a high performance computer containing a J-11 microprocessor with a floating-point processor. The CPU executes the PDP-11 instruction set. The system operates on Digital Equipment Corporation's 18-bit UNIBUS with a 22-bit memory addressing capability.

1.2 System Components

Figure 1-1 shows the system block diagram. The kernel system consists of:

- KDJ11-BF CPU module
- One MSV11-JD 1 Mbyte ECC memory module, one MSV11-JE 2 Mbyte ECC memory module, or two MSV11-JE 2 Mbyte ECC memory modules
- KTJ11-B UNIBUS adapter (UBA) module

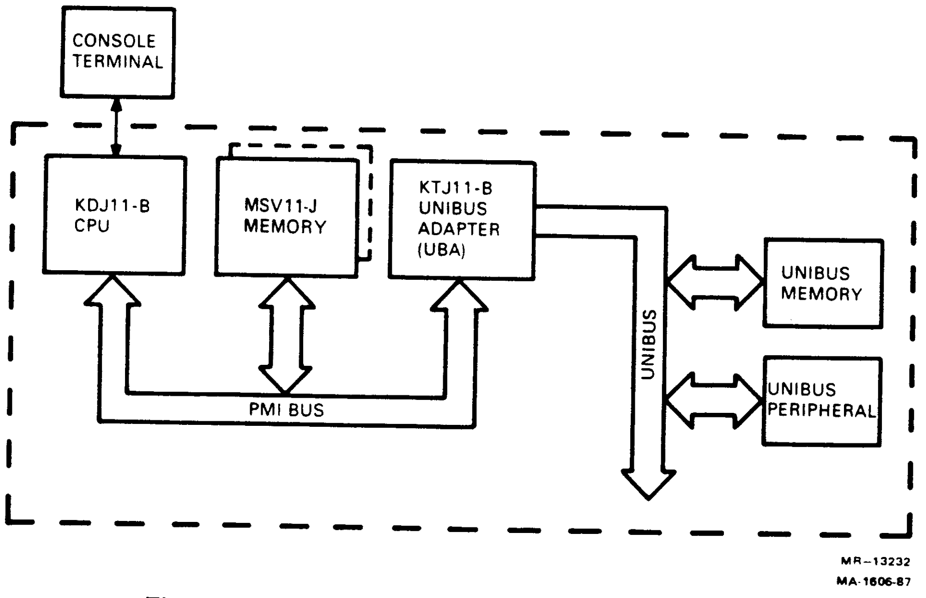

Figure 1-1 PDP-11/84 System Block Diagram

The modules communicate through the high-speed private memory interconnect (PMI) bus using 22-bit address and 16-bit data lines.

A console serial line unit supporting the EIA RS232 communication standard allows you to connect a console terminal to the KDJ11-BF CPU module.

The KTJ11-B UBA module interfaces to the PMI bus and the UNIBUS. The UBA module supports all address and data communications between the processor memory and all UNIBUS peripherals. In addition, the UBA module serves as a terminator for the CPU end of the UNIBUS.

1.2.1 KDJ11-BF CPU Module

The KDJ11-BF (M8190-AE) is a quad CPU module with the complete functionality of a PDP-11 processor. The KTJ11-B UNIBUS adapter module allows the CPU to interface with Digital's UNIBUS.

The module features a J-11 microprocessor, a floating-point processor, 22-bit memory management, an 8 Kbyte cache memory, a programmable line frequency clock, a console serial line unit, a configuration electrically erasable programmable read only memory (EEPROM), and boot and diagnostic ROMs.

In addition, the KDJ11-BF has six red LED indicators for displaying diagnostic information during power-up and bootstrapping. A single green LED indicates dc power to the module.

1.2.2 KTJ11-B UNIBUS Adapter Module

The KTJ11-B UNIBUS adapter (M8191) is a hex module that interfaces with the KDJ11-BF processor and memory through the PMI. The module contains the PMI adapter logic, UNIBUS mapping, four M9312 compatible boot ROM sockets, and a DMA cache.

1.2.3 MSV11-JD/JE Memory Module

The quad memory module is available in two versions:

- MSV11-JD (M8637-DA) with a 1 Mbyte capacity
- MSV11-JE (M8637-EA) with a 2 Mbyte capacity

The modules provide error correcting code (ECC) and support write byte/word, double word read, and block mode read operations over the PMI bus. A red LED indicates the occurrence of an uncorrectable error. A green LED indicates the presence of +5.1 VB power.

1.3 Related Documents

Table 1-1 lists some documents that contain information related to the PDP-11/84 or its operating systems.

Table 1-1 Related Documents

Title	Order Number
PDP-11/84-D User's and Maintenance Guide	EK-1184D-UG
PDP-11/84-E User's and Maintenance Guide	EK-1184E-UG
PDP-11/84-D Field Maintenance Print Set	MP-02536-01
PDP-11/84-E Field Maintenance Print Set	MP-01955-01
PDP-11/84-D Illustrated Parts Breakdown	EK-1184D-IP
PDP-11/84-E Illustrated Parts Breakdown	EK-1184E-IP
PDP-11 UNIBUS Processor Handbook	EB-26077-41
PDP-11 Architecture Handbook	EB-23657-18
KDJ11-B CPU User Guide	EK-KDJ1B-UG
MSV11-J Memory User Guide	EK-MSV1J-UG
DCJ11 Microprocessor User Guide	EK-DCJ11-UG
Supermicrosystems Handbook	EB-27713-41
Chipkit Handbook	EK-01387-92
Communications Handbook	EB-30066-42
Terminals and Printers Handbook	EB-23909-54
PDP-11 Software Handbook	EB-25398-41
RSX-11 Handbook	EB-25742-41
RSTS/E Handbook	EJ-23534-18
ULTRIX Software Guidebook	EJ-26153-20

1.3.1 Digital Personnel Ordering Information

Additional copies of this document and printed copies of the documents listed may be obtained from:

Digital Equipment Corporation
10 Forbes Road
Northboro, MA 01532
ATTN: Printing and Circulation Services (NRO3/A5)
Customer Services

1.3.2 Customer Ordering Information

Purchase orders for supplies and accessories should be sent to:

Digital Equipment Corporation
P.O. Box CS2008
Nashua, NH 03061

2 Functional Description

2.1 Introduction

The PDP-11/84 functional block diagram is shown in Figure 2–1. The three modules are the KDJ11-BF CPU, MSV11-JD/JE ECC memory, and KTJ11-B UNIBUS adapter (UBA). The KDJ11-BF module has a J-11 microprocessor with an integral floating-point processor, an 8 Kbyte cache memory, private memory interconnect (PMI) arbitration logic, memory management registers, EEPROM/ROM, configuration registers, a serial line unit (SLU), six red diagnostic LEDs, and a green +5 V power LED.

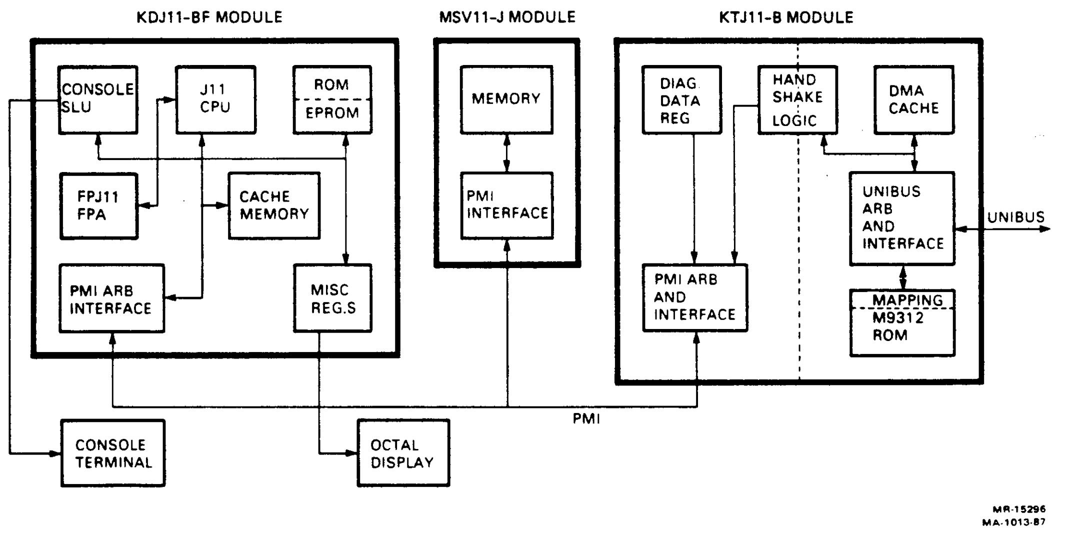

Figure 2–1 PDP-11/84 Functional Block Diagram

The KTJ11-B UNIBUS adapter module is divided into a PMI section and a UNIBUS section. The handshake logic enables data transfers to occur between PMI and UNIBUS devices. The PMI section has diagnostic registers and PMI arbitration/interface logic. The UNIBUS section has a 32-word DMA cache, UNIBUS mapping logic, sockets for M9312-type user ROMs, and UNIBUS arbitration and interface logic.

The MSV11-JD memory module has a 1 Mbyte capacity, while the MSV11-JE has a 2 Mbyte capacity. A maximum of two memory modules are allowed in the system configuration.

The modules transfer data using the PMI. Data transfers between the PMI and UNIBUS devices use the handshaking logic on the KTJ11-B module. PMI read operations (DATI, DATIP, and DATBI) can be word or block mode. PMI write operations (DATO, DATOB) can be word or byte mode.

NOTE

The PDP-11/84 UNIBUS power-up protocol is slightly different from most PDP-11 systems. See Appendix B for a protocol description.

All communications between UNIBUS devices and the UBA occur through standard UNIBUS protocol. No Q-bus devices may be configured on the system.

2.2 PMI Bus Description

The PMI bus provides a high-performance communication path between the KDJ11-BF CPU, the MSV11-JD/JE memory, and the KTJ11-B UBA. The PMI consists of 14 signals that are unique to PMI protocol. The signal lines are described in Table 2-1.

The KDJ11-BF CPU module is also used in systems that contain Q-bus devices. Some of the signals, therefore, retain their Q-bus names. The functionality of these signals changes, however, when a KTJ11-B UBA is part of the system. Data and address information is multiplexed and uses the same data/address lines as Q-bus protocol.

Table 2-1 PMI Signal Line Descriptions

Signal Line	Description
BDAL <21:00>	22 Multiplexed Bidirectional Data/Address Lines. During the address phase of a data transfer cycle, the PMI master gates address information onto these lines. During the data phase of the cycle, the slave (DATI) or the master (DATO) gates data onto BDAL <15:00> and parity error/control information onto BDAL <17:16>.
BBS7	Bank 7 Select (I/O Page Select). When the PMI master gates an address onto BDAL <21:00>, it asserts BBS7 to reference the I/O page (including the I/O page addresses reserved as nonexistent memory). When BBS7 is asserted, BDAL <12:00> specifies the I/O page address; Negation of BBS7 selects the memory address space.
BRPLY	Reply This signal is asserted by the UBA as a slave response during the PMI DATO(B) cycle and during the interrupt vector DATI cycle.
BDIN	Data Input This signal is used by PMI protocol during UNIBUS interrupt grant cycles. The CPU asserts BDIN after gating the interrupt priority onto BDAL <03:00>. The UBA latches the interrupt priority on the leading edge of BDIN.
BIACKI	Interrupt Acknowledge

Table 2-1 (Cont.) PMI Signal Line Descriptions

Signal Line	Description
	This signal is used by PMI protocol during UNIBUS interrupt grant cycles. When the UBA receives the assertion of BIACK (from the CPU), it asserts one of the UNIBUS grant (BGn) signals.
BPOK	Power OK This signal is asserted and negated by the UBA in response to AC LO on the UNIBUS following standard UNIBUS power-up/power-down protocol. UNIBUS devices and/or PMI memory may, during power-up, prolong the assertion of AC LO/BPOK.

The following signals are asserted (low) and negated (high) by the PMI master:

Signal Line	Description
PBCYC	PMI Bus Cycle The PMI master asserts this signal at the start of a PMI cycle and negates this signal at the end of that cycle.
PBYT BWTBT	PMI Byte and Write Indication When the PMI master gates an address onto BDAL <21:00>, it asserts or negates these signals to indicate what type of data transfer will occur during the next bus cycle:

BWTBT	PBYT	Bus Cycle Type
H	H	DATI or DATBI
H	L	DATIP
L	H	DATO
L	L	DATOB

Signal Line	Description
PBLKM	PMI Block Mode When a PMI master wants to read more than two words of data, it uses both PBCYC and PBLKM to control the timing of the block mode data in (DATBI) cycle. It asserts both PBCYC and PBLKM at the start of the DATBI cycle. It negates PBLKM after reading two data words and then reasserts PBLKM (unless the next two words will end the cycle). After reading the last two words, the PMI master negates PBCYC (PBLKM is already negated).
PWTSTB	PMI Write Strobe The PMI master asserts this signal after gating data onto BDAL <15:00>. The PMI slave latches the data into its write buffer on the leading edge of the PWTSTB pulse.
QSACK	The UBA asserts this signal on the PMI in response to SACK from the UNIBUS.
DMR	The UBA asserts this signal on the PMI in response to NPR from the UNIBUS or when the UBA is performing a DMA cycle in its own behalf.

Table 2-1 (Cont.) PMI Signal Line Descriptions

Signal Line	Description
DMG	The CPU asserts this signal when PMI master status has been granted to the UBA in response to a DMG.
QBR7-4	The UBA asserts one of these signals in response to one of the BR7-4 lines being asserted on the UNIBUS during interrupt request cycles.

The following signals are asserted and negated by the PMI slave:

PSSEL	PMI Slave Selected The PMI slave (CPU or memory only) asserts this signal whenever it decodes a valid address on BDAL <21:00>.

NOTE

When PUBMEM is asserted, the PMI slave does not respond to PMI control signals. PUBMEM is asserted by the UBA to indicate that UNIBUS memory space is being addressed. The UBA does not assert PSSEL. The CPU ignores the assertion of PSSEL if PUBMEM is asserted.

PHBPAR	PMI High Byte Data Parity This signal is generated by PMI memory during DATI and DATBI cycles and provides odd parity for the high byte data (on BDAL <15:08>).
PLBPAR	PMI Low Byte Data Parity This signal is generated by PMI memory during DATI and DATBI cycles and provides even parity for the low byte data (on BDAL <07:00>).
PRDSTB	PMI Read Strobe The PMI slave asserts and negates this line to control data transfers during DATI and DATBI cycles. The PMI master latches the first word of the received data on the negating edge of this signal. The PMI master latches the second data word a specified time after this signal is negated.
PSBFUL	PMI Slave Buffer Full A PMI slave asserts PSBFUL during a write cycle, indicating that its write buffer is full and that it cannot respond to another cycle request. The new PMI master may gate an address onto BDAL <21:00> while PSBFUL is asserted, but it must not assert PBCYC until PSBFUL is negated.

The following signals are used for communication between the CPU and the UNIBUS adapter. PMI memory modules do not use these signals.

PMAPE	PMI UNIBUS Map Enable The CPU module asserts this signal if memory management register 3 (MMR3) bit <5> is set and negates this signal if MMR <05> is clear. The UBA module enables the UNIBUS map if PMAPE is asserted and disables the UNIBUS map if PMAPE is negated.

Table 2-1 (Cont.) PMI Signal Line Descriptions

Signal Line	Description
PUBSYS	PMI UNIBUS System This signal is a static signal that indicates whether the system is a UNIBUS system or a Q-bus system and is asserted by the UBA. The CPU follows PMI protocol for data transfers whether PSSEL is asserted or not.
PUBMEM	PMI UNIBUS Memory This signal line is asserted by the UBA to indicate that the UNIBUS memory space is being addressed. The UBA asserts PUBMEM during the assertion of PBCYC. **NOTE** **When PUBMEM is asserted, the PMI slave does not respond to PMI control signals. PSSEL is asserted by PMI memory when addressed. The CPU ignores the assertion of PSSEL if PUBMEM is asserted. The UBA does not assert PSSEL.**
PUBTMO	PMI UNIBUS Timeout This signal is asserted by the UBA in response to any of the following conditions: • A nonexistent memory timeout occurs when the UBA sends an address out on the UNIBUS. • A SACK timeout occurs during an interrupt cycle. • An interrupting UNIBUS device has been granted UNIBUS master status but does not execute an interrupt transaction.
PBSY	PMI Busy This signal is asserted by the PMI master (CPU or UBA) when it gains PMI master status and is negated by the PMI master when it relinquishes PMI master status. The CPU is the PMI master on power-up.

2.2.1 PMI Bus Acquisition

In the PDP-11/84 system, the CPU is the default PMI bus master; the UBA is the default UNIBUS master. This is always the condition at power-up. When the UBA is not requesting the PMI bus, the CPU arbitrates to become PMI master and holds PBUSY asserted.

Unlike previous PDP-11 systems, when no device on the UNIBUS is requesting use of the bus, the UBA arbitrates to become UNIBUS master and holds BBSY asserted.

The CPU relinquishes PMI master status when responding to a DMA request or an interrupt cycle from the UBA. Once the CPU has relinquished control of the PMI, it can regain PMI master status only when the following conditions are met.

- QSACK has been negated for 75 ns minimum.
- PBSY has been negated for 0 ns minimum.

When the CPU, as PMI master, references a memory or I/O page address on the UNIBUS, the UBA responds as the slave on the PMI. The UBA continues to control the UNIBUS side of the data transfer as bus master.

The UBA becomes PMI master when the CPU issues a DMA grant (DMG) or performs an interrupt cycle. The UBA may accept the DMG or interrupt grant, thus becoming both PMI and UNIBUS master at the same time. Alternatively, the UBA may pass the DMG or interrupt grant on to a requesting UNIBUS device, which would then become UNIBUS master.

Master status of the UNIBUS and/or PMI bus is requested as follows:

- A UNIBUS device can become UNIBUS master through an NPR request and control data transfers over the UNIBUS. During these data transfers, the UBA is PMI master and responds as UNIBUS slave if the UNIBUS device accesses a PMI memory location, a PMI I/O page location, or a UBA I/O page location.
- A UNIBUS device can become UNIBUS master through a BR7-4 request. As UNIBUS master, the device can control data and/or interrupt vector transfers. In both cases the UBA will respond as UNIBUS slave. The device may perform an interrupt vector cycle or access a PMI memory location, a PMI I/O page location, or a UBA I/O page location.
- The UBA can become both PMI and UNIBUS master at the same time through DMG and interrupt requests. As PMI and UNIBUS master, the UBA has direct access to the PMI.

2.2.2 DMA Requests

Unlike previous PDP-11 system CPUs, the PDP-11/84 CPU does not necessarily give unconditional priority to DMA requests. It can be programmed (see Section 3.3 on setup mode) to give itself priority over DMA requests after waiting for this programmed length of time to perform a memory transfer or to honor an interrupt request.

Through an NPR request to the UBA, a UNIBUS DMA device can perform UNIBUS DATI, DATIP, DATO, and DATOB cycles. The UBA controls the PMI portion of the data transfer.

When placed in diagnostic test mode, the UBA can perform DMA transfers without a requesting UNIBUS device. In this case, the UBA itself is the requesting device. When the UBA performs a DMA cycle, it becomes master of the PMI and the UNIBUS simultaneously, and has complete control of the PMI.

The following PMI protocol flow is observed by the CPU and UBA when arbitrating a nonprocessor request from a UNIBUS device.

PMI Bus	UNIBUS
	1. The UNIBUS device asserts NPR.
	2. If the UBA is UNIBUS master, it negates BBSY after removing address, data, and control information from the bus.
3. The UBA asserts DMA request (DMR) on the PMI bus.	
4. The CPU bus arbitration logic asserts DMG on the PMI after receiving DMR and 75 ns minimum after the negation of QSACK from a previous PMI bus transaction.	
5. The UBA receives the assertion of DMG.	
	6. The UBA asserts nonprocessor grant (NPG).
	7. The requesting device with the highest priority asserts SACK and negates nonprocessor request (NPR).
8. The UBA asserts QSACK on the PMI.	
NOTE **The UBA asserts PUBTMO instead of QSACK (indicating no SACK timeout) if SACK is not received within 10 μs after it asserts BGn on the UNIBUS. The CPU then cancels the DMA cycle and resumes arbitration.**	
9. The CPU arbitration logic receives QSACK and negates DMG.	

PMI Bus	UNIBUS
NOTE **Because the UBA provides the No SACK Timeout function on the PMI, the CPU always asserts DMG until it receives QSACK or UBTMO.**	
	10. The UBA negates NPG on the UNIBUS.
11. The UBA asserts PBSY after receiving the negation of PBSY from the previous PMI cycle, and becomes PMI master.	
	12. After receiving the negation of BBSY from the previous bus master, the UNIBUS device asserts BBSY and negates SACK.

When a UNIBUS DMA device becomes UNIBUS master through an NPR request, it can perform UNIBUS DATI, DATIP, DATO, and DATOB cycles. If the device accesses a PMI memory location, a PMI I/O Page location, or a UNIBUS I/O page location, on the UBA, the UBA responds as the UNIBUS slave. For PMI memory and PMI I/O page accesses, the UBA—as the PMI master—controls the PMI portion of the data transfer.

Data transfer cycles are described in Section 2.2.5.

PMI Bus	UNIBUS
13. The UBA negates QSACK.	
14. The CPU resumes arbitration 75 ns minimum after receiving the negation of QSACK.	
	15. The UNIBUS device removes address, data, and control information from the bus and negates BBUSY.
16. After the PMI slave or the UBA has removed all data and control information from the bus, the UBA negates PBSY.	

2.2.3 UNIBUS Device Interrupt Requests

The CPU and UBA observe the following protocol flow when arbitrating interrupt requests.

PMI Bus	UNIBUS
	1. The UNIBUS device asserts the appropriate interrupt request line BR7-4.
2. The CPU receives the appropriate request level on QBR7-4.	
3. The CPU arbitration logic asserts one of the four lines, DAL <03:00>, to indicate the level of the granted interrupt.	
4. The CPU asserts BDIN 150 ns minimum after gating DAL <03:00> onto the bus.	
5. The CPU asserts BIAK 225 ns minimum after it asserts BDIN.	
6. The UBA latches DAL <03:00> on the asserting edge of BDIN.	
7. The CPU receives the assertion of IACK on the PMI.	

NOTE

The UBA compares the interrupt level being granted with its own interrupt level and can block the grant. In this case, the UBA performs an interrupt or data transfer cycle and has complete control of the PMI and the UNIBUS simultaneously. In this case, the BGn line would not be asserted on the UNIBUS.

PMI Bus	UNIBUS
	8. The UBA asserts the selected UNIBUS grant (BGn) line. DAL <03> = BG7, DAL <02> = BG6, etc.
	9. If the UBA was the UNIBUS master, it removes address, data, and control information from the bus and negates BBSY.
	10. The highest priority-requesting device receives the assertion of BGn and asserts SACK.
	11. The device negates its BRn.
12. The UBA asserts QSACK.	
NOTE **The UBA asserts PUBTMO (indicating No SACK Timeout) if SACK is not received on the UNIBUS within 10 μs after it asserts BGn. When the CPU receives the assertion of PUBTMO, it cancels the interrupt cycle and resumes arbitration.**	
	13. The UBA negates BGn.
	14. After receiving the negation of BBSY from the previous bus master, the UNIBUS device asserts BBSY and negates SACK.
15. After the UBA receives the negation of PBSY from the previous PMI cycle, the UBA asserts PBSY and negates QSACK.	
16. The UBA now has control of the PMI bus and may initiate a PMI data transfer cycle(s) and/or an interrupt cycle.	

NOTE

The CPU resumes NPR arbitration 75 ns after the negation of QSACK but does not resume BR arbitration until it has updated the PC and PSW to complete the interrupt cycle or has aborted the interrupt request.

When a UNIBUS device becomes UNIBUS master through an interrupt request, it can perform interrupt vector cycles or UNIBUS DATI, DATIP, DATO, and DATOB cycles. If the device accesses a PMI memory location, a PMI I/O page location, or a UNIBUS I/O page location, the UBA responds as UNIBUS slave. For PMI memory and PMI I/O page accesses, the UBA, as the PMI master, controls the PMI portion of the data transfer. BDIN and BIACK being asserted does not effect the data transfer.

Data transfer cycles are described in Section 2.2.5.

The following sequence describes the interrupt transfer cycle:

PMI Bus	UNIBUS
	17. The interrupting device, as bus master, gates its vector onto the data lines and asserts INTR.
18. The UBA, as PMI master, asserts BRPLY on the PMI.	
	19. The UBA as slave receives the assertion of INTR and latches the interrupt vector.
	20. The UBA asserts SSYN.
21. The UBA gates the vector onto the DAL lines.	
22. The CPU latches the interrupt vector 200 ns minimum after receiving BRPLY.	
23. The CPU negates BDIN and BIAK.	
24. The UBA receives the negation of BIACK and negates BRPLY.	
	25. After receiving SSYN, the device removes its vector from the data lines and negates INTR and BBSY.
26. The UBA negates PBSY.	

2.2.4 PMI Data Transfer Address Cycle

The addressing phase of the PMI cycle starts immediately after the CPU or UBA has gained PMI master status and has asserted PBSY on the PMI.

The PMI bus acquisition phase is described in Section 2.2.1.

PMI Master / **PMI Slave**

PMI Master

1. The address is gated out on BDAL <21:00>, and BS7 is asserted if the address is in the I/O page. The signal lines BWTBT and PBYT are asserted to indicate the cycle to be performed:

BWTBT	PBYT	Bus Cycle Type
H	H	DATI or DATB
H	L	DATIP
L	H	DATO
L	L	DATOB

PMI Slave

2. When a valid address is decoded by a slave, it responds as follows:
   a. The UBA asserts PUBMEM if the address is UNIBUS memory or UNIBUS I/O page.
   b. PMI memory and the CPU assert PSSEL.

PMI Master

3. PBCYC is asserted.
4. How the cycle proceeds is dependent on whether the CPU or UBA is master, and on what the response was from the slave as follows.
   a. When the CPU is PMI master:

      If PSSEL is asserted and PUBMEM is negated, the CPU proceeds with a PMI memory cycle.

      If PSSEL is negated, the CPU performs a PMI cycle with the UBA responding as the PMI slave.
   b. When the UBA is PMI master:

      If PSSEL is negated, then the UBA aborts the PMI cycle and does not respond as the UNIBUS slave.

2.2.5 PMI Data Transfer Protocol

Following the data transfer address cycle, the data transfer cycle begins. The transfer of data on the PMI can be grouped into three general types of PMI data transfer cycles:

- The data in (DATI) and data in pause (DATIP) cycles. These are used to read one or two words.
- The block data in (DATBI) cycle. This is used to read up to 16 words.
- The data out (DATO) and data out byte (DATOB). These cycles are used to write a single word or byte.

The following sections describe each of the data transfer cycles.

2.2.6 PMI Data In Cycles (DATI and DATIP)

When accessing the PMI memory address space, a PMI master uses the DATI(P) cycle to read either one or two words of data. When accessing either I/O page or UNIBUS memory, a PMI master reads single words only.

The PMI DATIP cycle is identical to the DATI cycle with one exception: PBYT is asserted with the address to indicate that the next cycle (immediately following the current cycle) will be a DATO cycle to the same address.

The following flow is a description of a DATI(P) data transfer cycle.

PMI Master	PMI Slave
PBCYC is asserted during the addressing phase of the cycle (described in Section 2.2.4).	
	1. Data from the specified address is gated onto the bus.
	2. If the slave is PMI memory, PHBPAR and PLBPAR are generated and gated onto the bus.
	NOTE **These parity bits are generated only for memory locations that are cached on the CPU (that is, PMI memory).**
	3. PRDSTB is asserted.
	4. PRDSTB is negated.

PMI Master	PMI Slave
5. The first data word, with PHBPAR and PLBAR, is latched on the negating edge of PRDSTB. If only one word is to be read, PBCYC is negated and the cycle ends. If two words are to be read, PBCYC remains asserted. If a read-modify-write (DATIP) is being performed, a DATO cycle will take place here. The DATO(B) cycle is described in Section 2.2.8.	
	6. The second data word is gated onto the bus 80 ns maximum after the negation of PRDSTB.
	7. PHBPAR and PLBPAR are generated for the second data word and are gated onto the bus 100 ns after the negation of PRDSTB.
8. The second data word, along with PHBPAR and PLBAR, are received 145 ns maximum after the negating edge of PRDSTB. PBCYC is negated and the cycle ends. If a read-modify-write (DATIP) is being performed, PBCYC remains asserted, and a DATO cycle is performed here. The DATO(B) cycle is described in Section 2.2.8.	
	9. Data is removed from the bus after receiving the negation of PBCYC.

2.2.7 PMI Block Mode Data In Cycles

When accessing the PMI memory address space, a PMI master uses the PMI block mode data in (DATBI) cycle to read up to 16 words of data. A PMI master does not use the DATBI cycle when accessing either the I/O page or UNIBUS memory.

The PMI master can start DATBI data transfers on even word boundaries only and does not cross 16 word boundaries. This means that at the transfer start, address bits <01> and <00> must both equal zero, and the master terminates the transfer when address bits <04:01> are all equal to one.

The following flow describes the DATBI cycle.

PMI Master	PMI Slave
PBCYC is asserted during the addressing phase of the cycle. The addressing phase is described in Section 2.2.4.	
1. PBLKM is asserted.	
	2. Data from the specified address is gated onto the bus.
	3. If the slave is PMI memory, PHBPAR PLBPAR are generated and gated onto the bus.
	NOTE **These parity bits are generated only for memory locations that are cached on the CPU (that is, PMI memory).**
	4. PRDSTB is asserted.
	5. PRDSTB is negated, and the data is removed from the bus.
6. The first data word, with PHBPAR and PLBAR, is latched on the negating edge of PRDSTB.	
	7. The second data word is gated onto the bus 80 ns maximum after the negation of PRDSTB.
	8. PHBPAR and PLBPAR are generated for the second word and gated onto the bus 100 ns maximum after the negation of PRDSTB.
9. The second data word, with PHBPAR and PLBPAR, is received 145 ns maximum after the negating edge of PRDSTB.	
10. PBLKM is negated after latching the second data word.	

PMI Master	PMI Slave
	11. The second data word is removed from the bus after receiving the negation of PBLKM.
12. Data transfer cycles continue in the same manner as steps 1 through 11 above until two words are left to be transferred. The last two words are transferred with the same timing, but the signal PBLKM is not asserted by the master.	
13. PBCYC is negated after latching the last data word.	
	14. Data is removed from the bus after receiving the negation of PBCYC.

2.2.8 PMI Data Out Cycles

The PMI master uses the PMI Data Out (DATO or DATOB) cycles to transfer a single word or byte to a PMI slave.

The following flow describes a DATO(B) cycle.

PMI Master	PMI Slave
PBCYC is asserted during the addressing phase of the cycle. The addressing phase is described in Section 2.2.4.	
1. PBCYC is asserted and data is gated onto the bus.	
2. PWTSTB is asserted.	
	3. The assertion of PWTSTB is received, and the data is latched in.
	4. PSBUFL is asserted.
5. PWTSTB is negated.	
6. PBCYC is negated.	
	7. PSBUFL is negated.

2.3 Memory Management

Memory management is located on the KDJ11-BF and is used to relocate a 16-bit virtual address, if necessary, and transmit the 22-bit physical address to cache memory, or PMI memory. The PMI function of memory management is address modification. The modification of addresses is called relocation because it consists of adding a fixed constant to a virtual address to create a physical address.

Using the process of relocation, a user can load a program into one area of physical memory and execute it as if it were located in another area of memory. Several user programs, for example, can be simultaneously stored in memory. When any one program is running, it must be accessed by the processor as if it were located in the set of addresses beginning at 0.

When the processor accesses virtual bus address 0, a base address is added to it. The relocated 0 location of the program is accessed. Typically, this base address is added to all references while the program is running. A different base address is used for each of the other programs in memory.

Memory management also allows the user to protect one section of memory from access by programs located in another section. Memory management divides memory into individual sections called pages.

Each page has a protection or access key associated with it. The key defines the type of access allowed on a particular page. With the memory management unit, a page can be keyed nonresident (memory neither readable nor writable), or for no write operations (memory readable). These two types of protection, in association with other features, enable the user to develop a secure operating system.

Memory management specifies relocation on a page basis. A large program can then be loaded into nonadjacent pages in memory. This capability eliminates the need to shuffle programs to accommodate a new one. It also minimizes unusable memory fragments, thus allowing more users to be loaded into a specific memory size.

A program and its data can occupy as many as 16 pages in the memory. The size of each page may vary and can be any multiple of 32 words up to 4096 words in length. This feature allows small areas of memory to be protected (stacks, buffers, etc.). In addition, the last page of a program, exceeding 4K words, can be of adequate length to protect and relocate the remainder of the program.

As a result, the memory fragmentation problem inherent with fixed-length pages is eliminated. The base address of each page can be any multiple of 32 words in the physical address space, thus ensuring efficient use of PMI memory. The variable page length also allows the pages to be dynamically changed at run time.

Memory management provides three separate sets of pages for use in the processor's kernel, supervisor, and user modes. These sets of pages increase system protection by physically isolating user programs from service supervisor programs and the kernel program.

The service programs are also separated from the kernel program. Separate relocation register sets greatly reduce the time necessary to switch context between mapping. The three sets of registers also aid the user in designing an operating system that has clearly defined communications, is modular, and is more easily debugged and maintained.

The virtual bus address space is further divided, within each of the kernel, supervisor, and user pages, into instruction space and data space (I and D space). I space contains code, that is, any word that is part of the program such as instructions, index words, and immediate operands. D space contains information that can be modified, such as data buffers.

By using this feature, memory management can relocate data and instruction references with separate base address values. It is possible, therefore, to have a user program of 64K words consisting of 32K of instructions and 32K of data.

The memory management registers consist of 48 page address registers (PARs), 48 page descriptor registers (PDRs), and four memory management registers (MMR0-3). These registers are located on the KDJ11-BF module. The following sections describe each of the registers.

2.3.1 Page Address Registers

The page address registers contain the 16-bit page address field (PAF). The PAR specifies the base address of the page (Figure 2-2). All bits are read/write. These registers are not affected by console start or a RESET instruction. Their state at power-up is undefined.

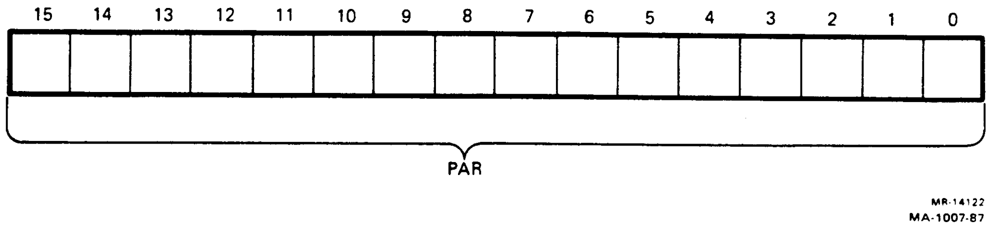

Figure 2-2 Page Address Register Format

2.3.2 Page Descriptor Registers

The page descriptor registers (PDRs) contain information relative to page expansion, page length, and access control. These registers are not affected by console start or a RESET instruction. Their state at power-up is undefined. All unused bits are read as zero and cannot be written. The register format is shown in Figure 2-3; bit descriptions are provided in Table 2-2.

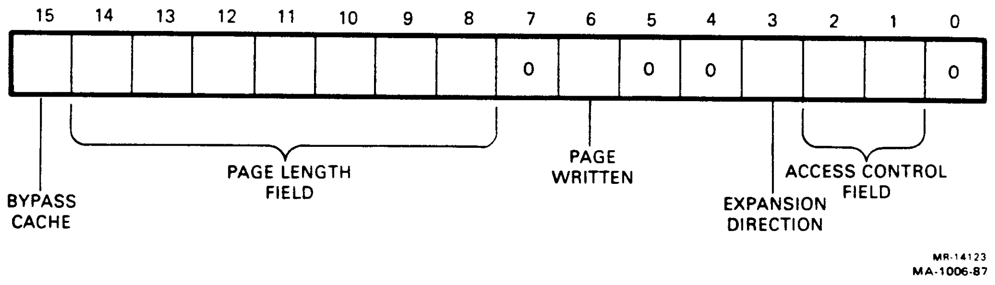

Figure 2-3 Page Descriptor Register Format

Table 2-2 Page Descriptor Register Bit Descriptions

Bit(s)	Name	Function
15	Bypass Cache (R/W)	This bit implements a conditional cache bypass mechanism. If set, references to the selected virtual page will bypass the cache. A cache bypass causes the cache location to be invalidated whenever a read or write hit occurs.
14:8	Page Length Field (R/W)	This field specifies the block number that defines the boundary of the current page. The block number of the virtual address is compared against the page length field to detect length errors. An error occurs when expanding upwards if the block number is greater than the page length field; an error occurs when expanding down if the block number is less than the page length field.
06	Page Written Field (R/W)	This bit indicates whether or not this page has been modified (written into) since either the PAR or PDR was loaded (1 is affirmative). It is useful in applications which involve disk swapping and memory overlays. It is used to determine which pages have been modified and hence must be saved in their new register bit and which pages have not been modified and can simply be overlaid. This bit is reset to 0 whenever either the PDR or the associated PAR is written into.
03	Expansion Direction (R/W)	This bit specifies in which direction the page expands. If ED=0, the page expands upwards from block number 0 to include blocks with higher addresses; if ED=1, the page expands downwards from block number 127 to include blocks with lower addresses. Upward expansion is usually used for program space while downward expansion is used for stack space.
2:1	Access Control	This field contains the access rights to this particular page. The access codes, or "keys," 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. The access codes are: 00 Nonresident - abort all accesses 01 Read only - abort on writes 10 Not used -abort all accesses 11 Read/write

2.3.3 Memory Management Register 0

Memory management register 0 (MMR0), at address 17777572, contains error flags, the page number whose reference caused the abort, and various other status flags. MMR0 is cleared at power-up, by a console start and by a RESET instruction. Figure 2-4 shows the register format. Table 2-3 contains the bit descriptions.

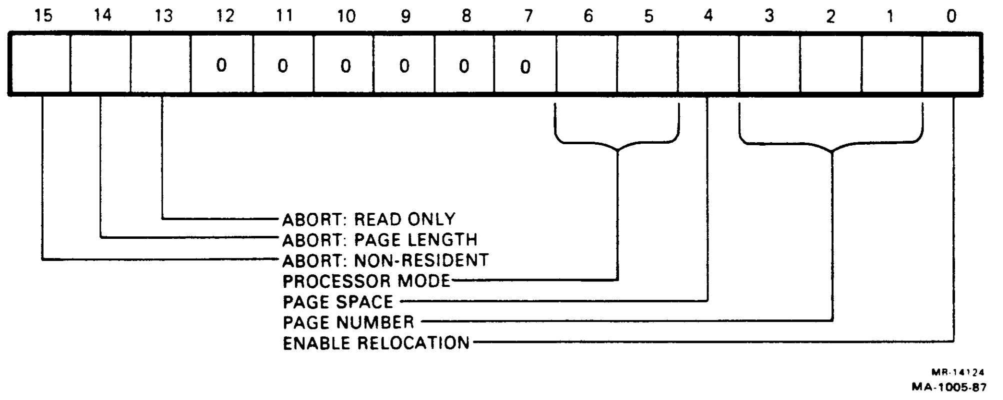

Figure 2-4 Memory Management Register 0 Format

Table 2-3 Memory Management Register 0 Bit Descriptions

Bit(s)	Name	Function
15	Abort—Nonresident (R/W)	Bit 15 is set by attempting to access a page with an access control field key equal to 0 or 2. It is also set by attempting to use memory relocation with a mode (PS <15:14>) of 2.
14	Abort—Page Length (R/W)	This bit 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 of the page descriptor register for that page.
13	Abort—Read Only (R/W)	This bit is set by attempting to write in a "read only" page. Read only pages have access keys of 1.

NOTE

Bits <15:13> can be set by an explicit write. This action, however, does not cause an abort. Whether set explicitly or by an abort, bits <15:13> cause memory management to freeze the contents of MMR0 <6:1>, MMR1, and MMR2. The status registers remain frozen until MMR0 <15:13> is cleared by an explicit write or any initialization sequence.

Bit(s)	Name	Function
6:5	Processor Mode (RO)	These bits indicate the processor mode (kernel/supervisor/user /illegal) associated with the page causing the abort (kernel = 00, supervisor = 01, user = 11, illegal = 10). If the illegal mode is specified, an abort is generated and bit <15> is set.
4	Page Space (RO)	This bit indicates the address space (I or D) associated with the page causing the abort (0 = I space, 1 = D space).
3:1	Page Number (RO)	These three bits contain the page number of the page causing the abort.

Table 2-3 (Cont.) **Memory Management Register 0 Bit Descriptions**

Bit(s)	Name	Function
00	Enable Relocation (R/W)	This bit allows address relocation. When set to 1, all addresses are relocated. When bit 0 is set to 0, memory management is inoperative and addresses are not relocated.

2.3.4 Memory Management Register 1

Memory management register 1 (MMR1) at address 17777574 records any auto increment or decrement of the general purpose registers. This register supplies the information necessary to recover from a memory management abort. MMR1 is read only. Its state at power-up is undefined. Figure 2-5 shows the register format.

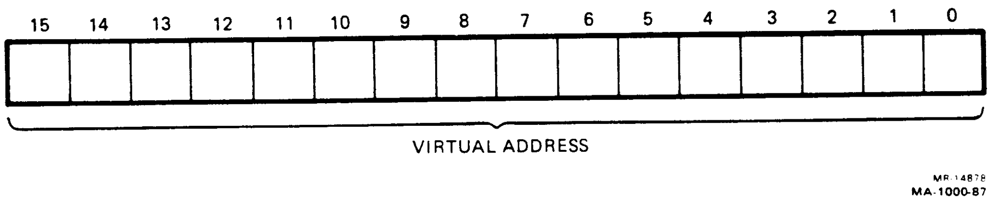

Figure 2-5 Memory Management Register 1 Format

2.3.5 Memory Management Register 2

Memory management register 2 (MMR2) at address 17777576 is loaded with the virtual address at the beginning of each instruction fetch. MMR2 is read only. Its state at power-up is undefined. Figure 2-6 shows the register format.

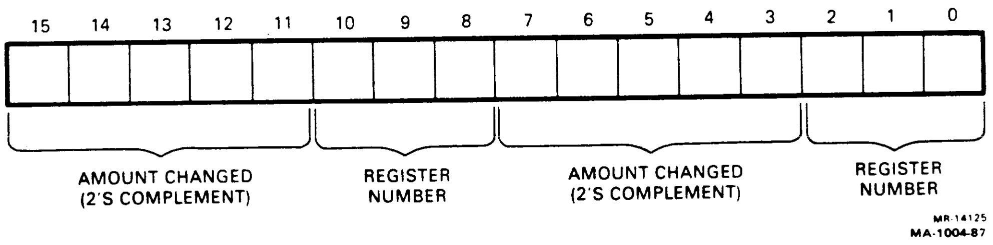

Figure 2-6 Memory Management Register 2 Format

2.3.6 Memory Management Register 3

Memory management register 3 (MMR3) at address 17772516 enables or disables D space, 22-bit mapping, the CSM instruction, and the I/O map (when applicable). MMR3 is cleared at power-up by a console start and by a RESET instruction. Figure 2-7 shows the register format. Table 2-4 provides the bit descriptions.

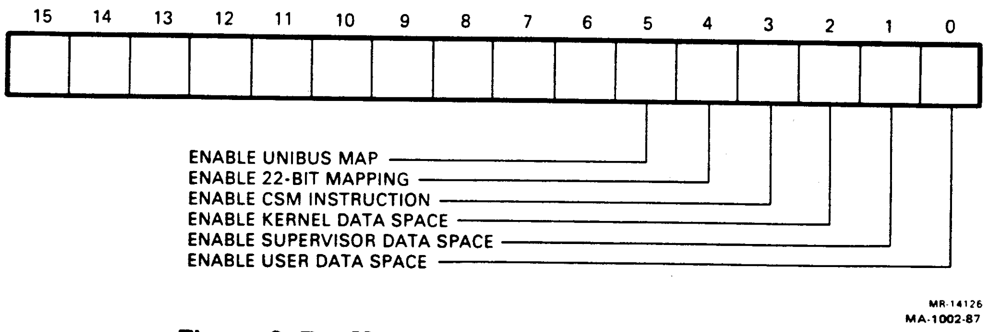

Figure 2-7 Memory Management Register 3 Format

Table 2-4 Memory Management Register 3 Bit Descriptions

Bit(s)	Name	Function
15:06	-	Unused.
05	Enable UNIBUS Map (R/W)	This bit enables the I/O map for the UNIBUS adapter.
04	Enable 22-bit Mapping (R/W)	This bit, when set, selects 22-bit memory addressing. When this bit is clear, 18-bit addressing is selected. (Only when MMR0 bit <0> is set is 18- or 22-bit addressing actually enabled.)
03	Enable CSM Instruction (R/W)	This bit enables recognition of the Call Supervisor Mode (CSM) instruction.
2:0	Enable Data Space (R/W)	These three bits enable data space mapping for kernel, supervisor, and user mode, respectively.

2.3.7 Physical Address Construction

If UNIBUS map relocation is enabled (MMR3 bit <05> = 1), UNIBUS address bits <17:13> select one of 31 mapping register pairs (corresponding to octal codes 00 through 36). The content of the selected mapping register pair is added to UNIBUS address bits <12:00> to produce the memory address. If UNIBUS address bits <17:13> are all 1s (octal code 37), the I/O page is selected. Memory address bits <21:18> are all set equal to zero, memory address bits <17:00> are identical to UNIBUS address bits <17:00>, and BBS7 is asserted (Figure 2-8).

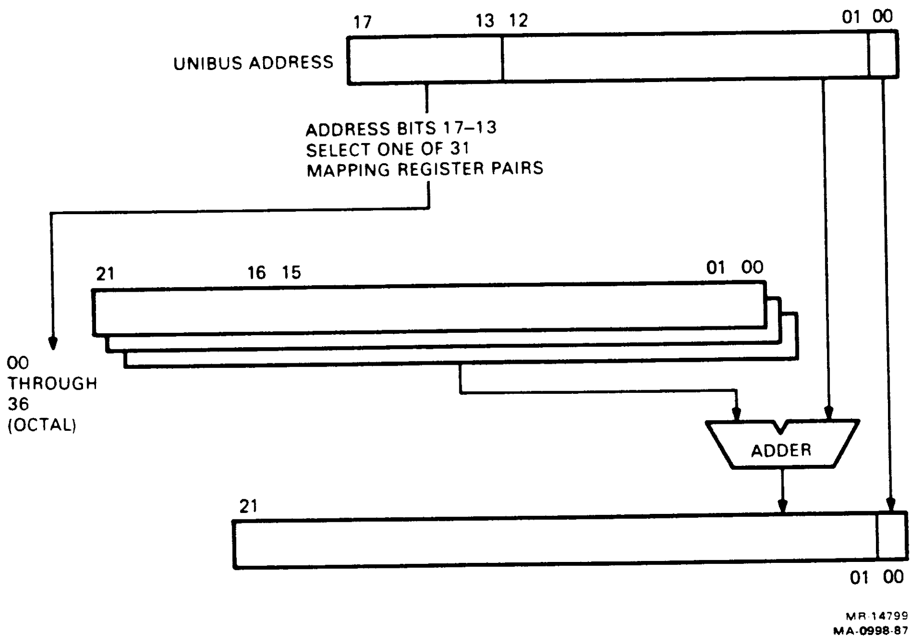

Figure 2–8 Physical Address Interpretation

2.3.8 Memory Relocation

When memory management is enabled, the normal 16-bit direct-byte address is no longer interpreted as a direct physical address. Instead, this address is interpreted as a virtual bus address containing information to be used in constructing a new 22-bit physical address. The information contained in the virtual bus address is combined with relocation information contained in the page address register to make a 22-bit physical address. Using memory management, memory can be dynamically allocated in pages composed of from 1 to 128 blocks of 32 words each.

The starting physical address for each page is a multiple of 32 words. Each page has a maximum size of 4096 words. Pages may be located anywhere within the physical address space. The set of 16 PARs to be used to create the physical address is determined by the current mode of operation of the CPU (kernel, supervisor, or user modes; see Section 2.10).

2.4 KDJ11-BF Cache

The CPU has dual tag cache memories. They are used to allow concurrent DMA activity and to decrease system access time of instructions and data. The 8 Kbyte cache is located on the KDJ11-B module. Cache operations occur only for PMI memory cycles; UNIBUS memory is not cached.

Figure 2–9 is a matrix showing the cache response for both DMA and CPU data transfers from and to the PMI memory space. Two cache memory tags are referenced in the matrix. The DMA matrix heading refers to DMA activity. The CPU matrix heading refers to PMI activity involving the CPU tag.

	DMA		CPU	
	HIT	MISS	HIT	MISS
READ	READ MEMORY	READ MEMORY	READ CACHED DATA	READ MEMORY AND ALLOCATE CACHE
WRITE WORD	INVALIDATE CACHE-WRITE MEMORY	WRITE MEMORY	WRITE BOTH CACHE AND MEMORY	WRITE BOTH CACHE AND MEMORY
WRITE BYTE	INVALIDATE CACHE-WRITE MEMORY	WRITE MEMORY	WRITE BOTH CACHE AND MEMORY	WRITE MEMORY –NO CACHE CHANGE
READ BYPASS	N/A	N/A	INVALIDATE CACHE AND READ MEMORY	READ MEMORY –NO CACHE CHANGE
WRITE BYPASS	N/A	N/A	INVALIDATE CACHE AND WRITE MEMORY	WRITE MEMORY –NO CACHE CHANGE
READ FORCE MISS	READ MEMORY	READ MEMORY	READ MEMORY –NO CACHE CHANGE	READ MEMORY –NO CACHE CHANGE
WRITE FORCE MISS	WRITE MEMORY-INVALIDATE CACHE	WRITE MEMORY	WRITE MEMORY –NO CACHE CHANGE	WRITE MEMORY –NO CACHE CHANGE

MR-14921
MA-0990-87

Figure 2–9 Cache Response Matrix

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

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

2.4.1 KDJ11-BF Cache Operations

The KDJ11-BF cache is initially flushed (emptied). As the KDJ11-BF addresses PMI memory locations, the KDJ11-BF cache begins to fill with addresses and data. If the KDJ11-BF addresses PMI memory within an 8 Kbyte memory space, the cache fills, as the addresses are accessed, to the 8 Kbyte limit of the cache size. This means that for each cache address location the data for that address is a duplicate of the corresponding PMI address's data.

As each instruction is accessed by the KDJ11-BF, its address is compared in cache to see if there are any matches. If a match occurs, the PMI memory cycle is aborted and the data stored in the cache address is used by the KDJ11-BF. If a cache miss occurs, the PMI memory cycle is completed. In addition to filling its memory space, the cache monitors the PMI address lines for DMA writes to PMI memory addresses. If a write into a PMI memory address matches a cache address, that particular cache address is invalidated.

The following cache options are available on the KDJ11-BF.

- Conditional cache bypass—selected virtual pages can be made to bypass the cache. Bit <15> in the PDRs sets this condition.
- Unconditional cache bypass—all CPU references can be made to bypass the cache. Bit <9> in the cache control register sets this condition.
- Flush cache—all valid bits in the cache are cleared.
- Lock instruction (ASRB, TSTSET, and WRTCLK)—these instructions guarantee a cache bypass reference.

2.4.2 KDJ11-BF Cache Organization

The KDJ11-BF contains an 8 Kbyte direct map cache with dual tag store which allows concurrent operation of the CPU and DMA. The cache memory can be subdivided into three distinct sections: data store, CPU tag store, and the DMA tag store.

The KDJ11-BF cache interprets the CPU (or DMA) physical address as shown in Figure 2-10. Table 2-5 contains the bit description.

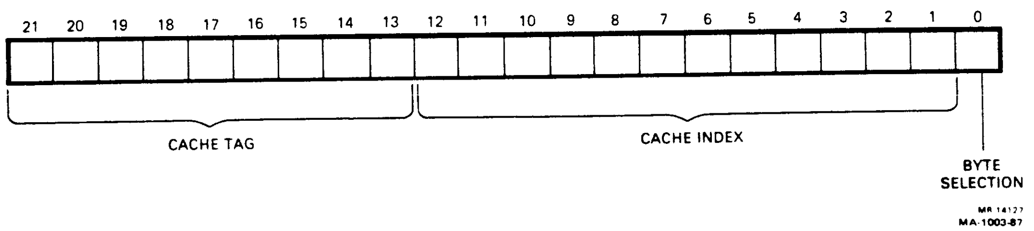

Figure 2-10 CPU/DMA Physical Address Interpretation Register

Table 2-5 CPU/DMA Physical Address Interpretation Bit Descriptions

Bit(s)	Name	Function
21:13	Cache Tag (R/W)	1. During CPU read/write operations, these bits are compared with bits <21:13> of the CPU cache tag register (Figure 2-11) to determine the cache hit/miss status. 2. During DMA read/write operations, these bits are compared with bits <21:13> of the DMA tag register (Figure 2-12) to determine the cache hit/miss status. For either CPU or DMA operations, a tag hit occurs when the cache tag contents match the CPU/DMA tag register and the CPU/DMA valid bit is set.
12:01	Cache Index (R/W)	The CPU cache interprets the CPU/DMA physical address directly and selects one of 4096 word cache memory locations.
00	Byte Selection	During CPU/DMA write operations setting, this bit selects writing into the high-byte cache memory location (Figure 2-13).

The high-byte parity bit reflects odd parity on data bits <15:08>.

The low-byte parity bit reflects even parity on data bits <07:00>.

The CPU tag parity bit reflects odd parity on CPU tag bits <21:13>.

The DMA tag parity bit reflects odd parity on DMA tag bits <21:13>.

The CPU and DMA tag valid bits are not included in the CPU and DMA tag parity calculations.

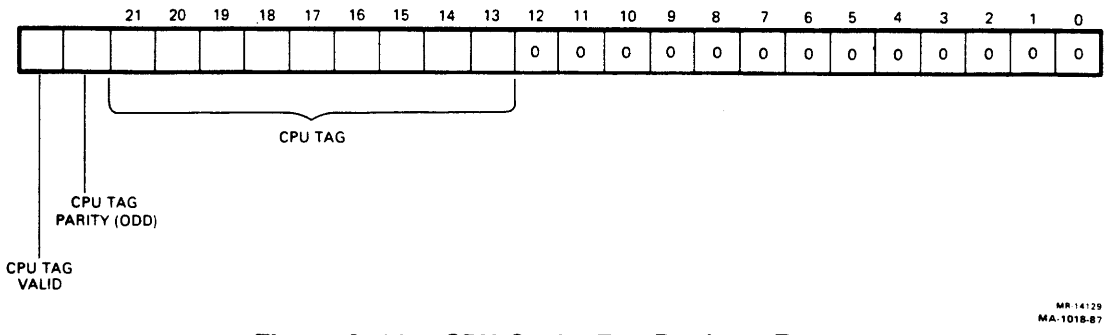

Figure 2-11 CPU Cache Tag Register Format

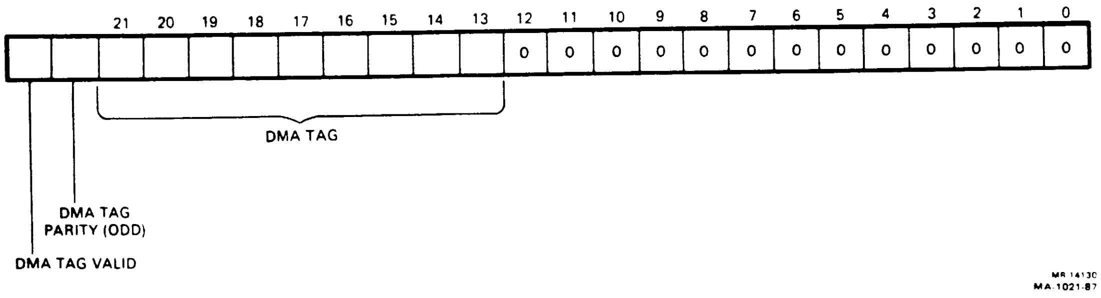

Figure 2-12 DMA Tag Register Format

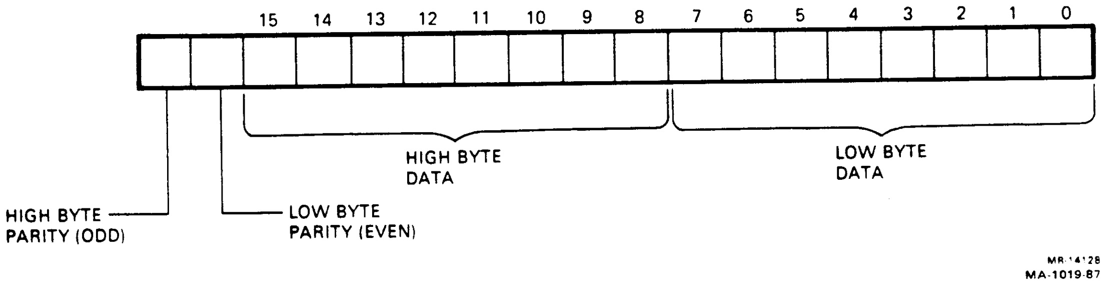

Figure 2-13 CPU Cache Data Organization

2.4.3 Cache Control Register

The cache control register (CCR) at address 17777746, controls the operation of the cache. Two copies of the cache control register are kept on the KDJ11-BF. One copy is kept in the chip set, the other on the board. The chip copy implements bits <10:0> as read/write but interprets only bits <9> and <3:2>. This copy is used as the source of data when the register is read. The board copy implements bits <10,8,7,6,0>. This is a write-only copy. It cannot be explicitly accessed. Figure 2-14 shows the register format. Table 2-6 contains the bit descriptions.

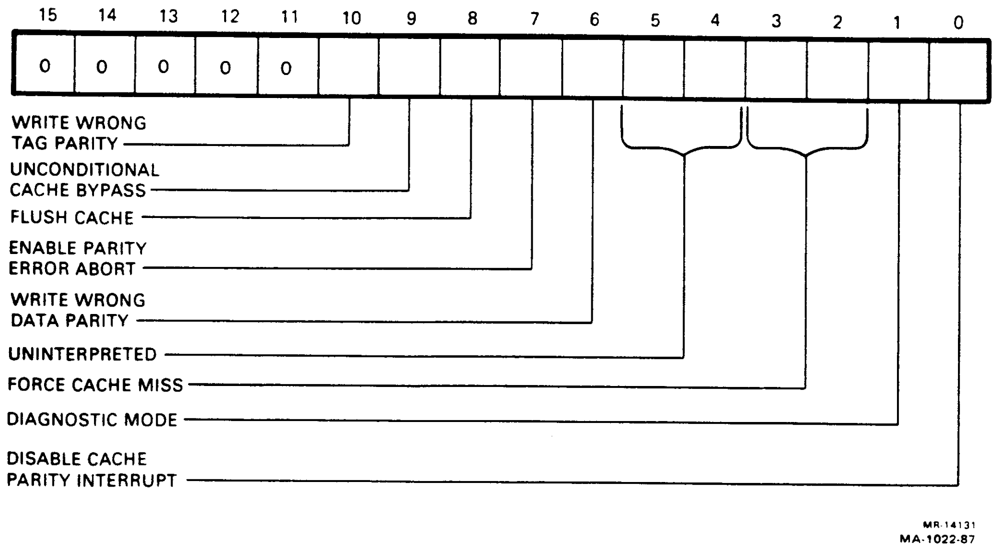

Figure 2–14 Cache Control Register Format

Table 2–6 Cache Control Register Bit Descriptions

Bit(s)	Name	Function
15:11	-	Unused. Always read as cleared bits.
10	Write Wrong Tag Parity (R/W)	When this bit is set, the CPU and DMA tag parity bits are both written as wrong parity during all operations that update these bits. A cache tag parity error will thus occur on the next access to that location.
09	Unconditional Cache Bypass (R/W)	When this bit is set, all references to memory by the CPU will bypass the cache and go directly to main memory. Read or write hits will result in the invalidation of the corresponding cache location; misses will not affect the cache contents.
08	Flush Cache (WO)	Writing a 1 into this bit clears all CPU tag and DMA tag valid bits invalidating the entire contents of the cache. Writing a 0 into this bit has no effect. Flush cache always reads as 0. The KDJ11-BF requires approximately 1 ms to flush the cache. During the period, DMA activity is possible and CPU activity is suspended.
07	Enable Parity Error Abort (R/W)	This bit is set for diagnostic purposes only. When it is set, a cache parity error (during a CPU cache read) will cause the CPU to abort the current instruction and trap to parity error vector 114. When this bit is clear, a cache parity error (during a CPU cache read) results in a force miss and data fetch from main memory. The CPU will trap to 114 only if CCR bit <0> is clear. DMA cycle cache parity errors will cause a trap to 114 if CCR <7> is set or if CCR <0> is clear. CCR <7> has no effect on main memory parity errors which always cause the CPU to abort the current instruction and trap to 114.

Table 2-6 (Cont.) Cache Control Register Bit Descriptions

Bit(s)	Name	Function
06	Write Wrong Data Parity (R/W)	When this bit is set, both the high and low data parity bits are written with wrong parity during all operations which update these bits. This will cause a cache data parity error to occur on the next access to that location.
03:02	Force Cache Miss (R/W)	When either of these bits is set, CPU reads will be reported as cache misses.
01	Diagnostic Mode (R/W)	When this bit is set, a 10 μs nonexistent memory timeout during a word write will not cause a nonexistent memory trap and will not set CPU error register bit <05>. All nonbypass and nonforced miss word writes will allocate the cache regardless of the nonexistent memory timeout.
00	Disable Cache Parity Interrupt (R/W)	This bit controls cache parity interrupts when CCR <7> is clear (normal operation). If CCR <7> is clear, a cache parity error (during a CPU cache read) results in a force miss and data fetch from main memory. The CPU will trap to 114 only if CCR bit <0> is clear. DMA cycle cache parity errors will cause a trap to 114 if CCR <7> is set or if CCR <0> is clear.

Table 2-7 summarizes the effect of CCR <7>, <0> on parity errors during CPU cache reads.

Table 2-7 Cache Parity Errors During CPU Cycles

CCR <7>	CCR <0>	Result of Cache Parity Error
0	0	Cache miss and update cache; interrupt to 114.
0	1	Cache miss and update cache; no interrupt.
1	X	Abort instruction and trap to 114.

Table 2-8 summarizes the effect of CCR <7>, <0> on DMA tag parity errors during DMA writes.

Table 2-8 Cache Parity Errors During DMA Cycles

CCR <7>	CCR <0>	Result of Cache Parity Error
0	0	Interrupt to 114.
0	1	No interrupt.
1	X	Trap to 114.

The CCR is cleared by the negation of DCOK and by a console start. It is not affected by BUS INIT. The J-11 ODT command G causes cache to be flushed and CCR to be cleared.

2.4.4 Memory System Error Register

The memory system error register (MSER) at address 17777744 reflects the status of cache and main memory parity errors. MSER bits <14> and <13> are used by the KDJ11-BF boot and diagnostic programs to test the cache DMA tag store. Figure 2-15 shows the register format; Table 2-9 contains the bit descriptions.

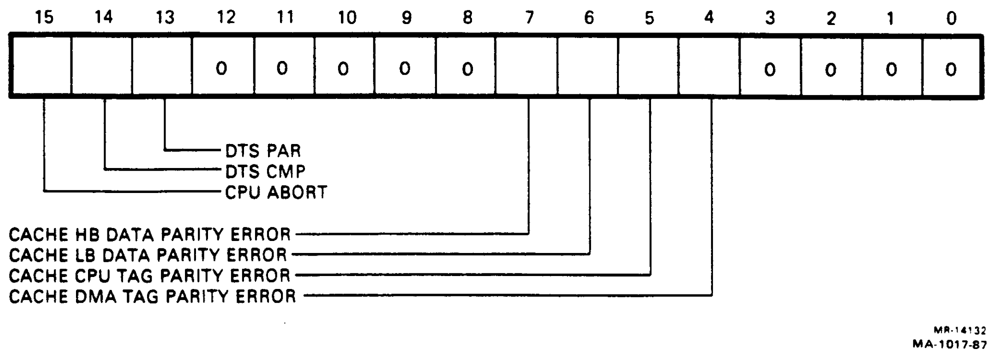
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
0
0
0
0
0
0
0
0
0
DTS PAR
DTS CMP
CPU ABORT
CACHE HB DATA PARITY ERROR
CACHE LB DATA PARITY ERROR
CACHE CPU TAG PARITY ERROR
CACHE DMA TAG PARITY ERROR
MR-14132
MA-1017-87

Figure 2-15 Memory System Error Register Format

Table 2-9 Memory System Error Register Bit Descriptions

Bit(s)	Name	Function
15	CPU Abort(RO)	This bit is set if a cache or main memory parity error results in an instruction abort (only during the demand read cycle). Cache parity errors cause an abort only if CCR <7> is set. Main memory parity errors always cause an abort.
14	DMA Tag Store Comparator (DTS CMP) (RO)	In standalone mode (BCSR <8> set), this bit indicates the output of the cache DMA tag store comparator for the previous non-I/O page reference with cache miss. When BCSR <8> is clear, DTS CMP reads as a 0.
13	DMA Tag Store Parity	(DTS PAR) (RO)
12:08	Unused	These bits always read as 0.
07	Cache HB Data Parity Error (R/W)	This bit is set if a parity error is detected in the high data byte during a CPU cache read. If CCR <7> is clear, MSER <7> is also set by a low byte parity error and by the set condition of MSER <5> or <4>.
06	Cache LB Data Parity Error (RO)	This bit is set if a parity error is detected in the low data byte during a CPU cache read. If CCR <7> is clear, MSER <6> is also set by a high byte parity error and by the set condition MSER bits <5> or <4>.
05	Cache CPU Tag Parity Error (RO)	This bit is set if a parity error is detected in the CPU tag field during a CPU cache read. If CCR <7> is clear, MSER <7> is also set by a high or low data byte parity error.

Table 2-9 (Cont.) Memory System Error Register Bit Descriptions

Bit(s)	Name	Function
		NOTE **Cache parity errors are ignored (do not affect MSER <7:5>) if either CCR <3> or <2> (force miss) is set or if the CPU tag valid bit is clear.**
04	Cache DMA Tag Parity Error (RO)	This bit is set if a parity error is detected in the DMA tag field during a DMA write operation.
		NOTE **Cache parity errors are ignored (do not affect MSER <4>) if either CCR <3> or <2> (force miss) is set or if the DMA tag valid bit is clear.**
03:00	-	Unused. These bits always read as 0.

Main memory parity errors always cause the CPU to abort the current instruction, to set MSER <15>, and to trap through vector location 114.

Cache parity errors that occur during a CPU cache access may result in an instruction abort and/or a trap to location 114, depending on the following condition of CCR bits <7> and <0>.

1. If CCR <7> (parity error abort) is set, a cache parity error causes the CPU to abort the current instruction, to set MSER <15> and the relevant error bit(s) MSER <7:5>, and to trap through vector location 114.
2. If CCR <7> is clear, and if CCR <0> is also clear, a cache parity error causes the CPU to force a cache miss, set the relevant error bits MSER <7:5>, and to trap through vector location 114.
3. If CCR <7> is clear, and if CCR <0> is set, a cache parity error causes the CPU to force a cache miss and to set the relevant error bits MSER <7:5>. The CPU does not trap through vector location 114.

Cache DMA tag field parity errors that occur during a DMA cycle cause a trap to location 114 if CCR <7> is set or if CCR <0> is clear.

The memory system error register is cleared by any MSER write reference. It is also cleared on power-up or by a console start. It is unaffected by a RESET instruction.

2.4.5 Hit/Miss Register

This register, at address 17777752, indicates whether the six most recent CPU memory references resulted in cache hits or cache misses. The hit/miss register is read only. Its value at power up is undefined. The hit/miss register is not affected by console start or a RESET instruction. The hit/miss register will always read 0 when the CPU is in console ODT mode. Figure 2-16 illustrates the register format. Table 2-10 contains the bit descriptions.

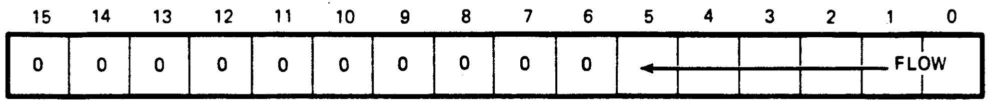

Figure 2-16 Hit/Miss Register Format

Table 2-10 Hit/Miss Register Bit Descriptions

Bit(s)	Name	Function
15:06	-	Unused. Always read as 0s.
05:00	Cache Hit	Bits enter from the right (at bit <0>) and are shifted left. A set bit indicates a cache hit; a cleared bit indicates a cache miss.

2.5 Additional CPU Register Descriptions

The general CPU registers include:

- Two sets of six working registers (R0–R5)
- Kernel/supervisor/user stack pointers (R6)
- Program counter (R7)

Other major registers are described in the following sections.

2.5.1 Processor Status Word

The processor status word (PSW) at location 17777776 contains information on the status of the processor. The PSW is initialized at power-up (depending on EEPROM CONFIGURATION options) and is cleared at console start. The RESET instruction does not affect the PSW. Figure 2-17 illustrates the register and Table 2-11 contains the bit descriptions.

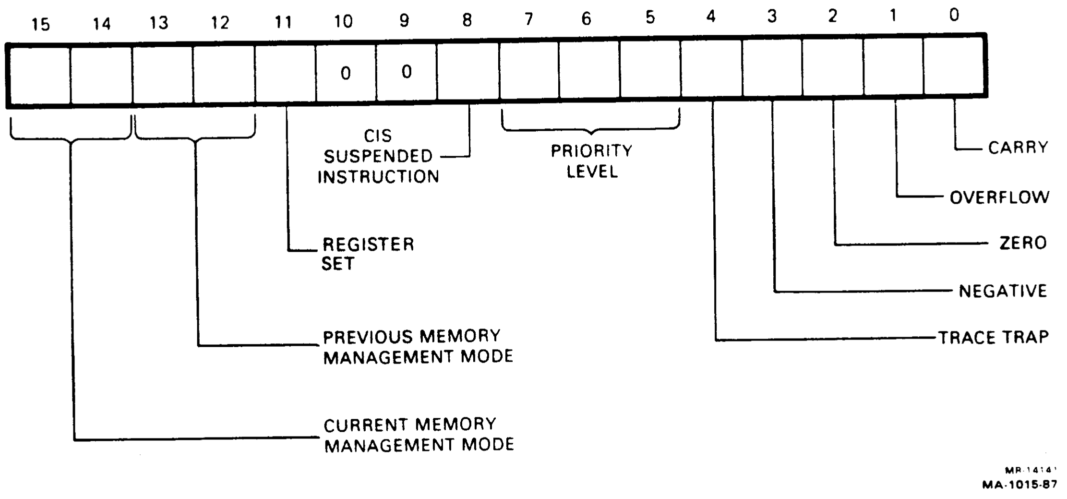

Figure 2-17 Processor Status Word Register

Table 2-11 Processor Status Word Bit Descriptions

Bit(s)	Name	Function
15:14	Current Mode (R/W, protected)	Current processor mode: 00 = kernel 01 = supervisor 10 = illegal (traps) 11 = user
13:12	Previous Mode (R/W, protected)	Previous processor mode, same encoding as current mode.
11	Register Set (R/W, protected)	General register set select: 0 = register set 0 1 = register set 1
08	Suspended Instruction (R/W)	Reserved for future use.
7:5	Priority (R/W, protected)	Processor interrupt priority level.
04	Trace Trap (R/W, protected)	Set to force a trace trap.
3:0	Condition Codes (R/W)	Processor condition codes.

2.5.2 Program Interrupt Request Register

The program interrupt request register (PIRQ), at location 17777772, implements a software interrupt facility. When a program interrupt request is granted, the processor traps through location 240. It is the responsibility of the interrupt service routines to clear the appropriate bit in PIRQ before exiting. PIRQ is cleared at power-up by a console start and by the RESET instruction. Figure 2-18 illustrates the register and Table 2-12 contains the bit descriptions.

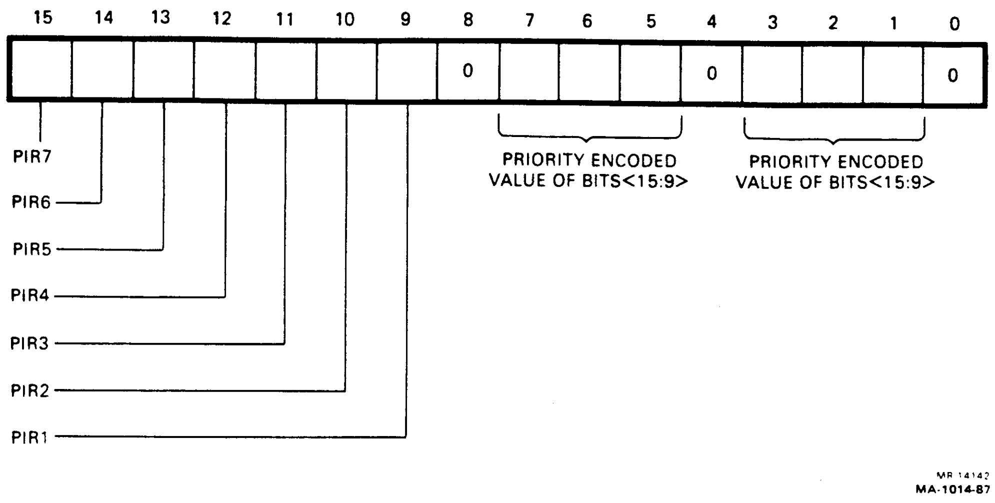

Figure 2-18 Program Interrupt Request Register

Table 2-12 Program Interrupt Register Bit Descriptions

Bit(s)	Name	Function
15:09	PIR 7-1	Each bit, when set, provides one of seven levels of software interrupt, corresponding to interrupt priority levels 7 through 1.
08	-	Unused
07:05	Priority encoded value of bits <15:09>	These three bits are set by the CPU to the encoded value of the highest pending interrupt request (bits <15:09>).
04	-	Unused
03:01	Priority encoded value of bits <15:09>	The function of these bits is identical to bits <07:05>.

2.5.3 CPU Error Register

The error register, at address 17777766, identifies the source of any abort or trap that caused a trap through location 4. The CPU error register is cleared when it is written. It is also cleared at power-up or by console start. It is unaffected by a RESET instruction. Figure 2-19 shows the register format and Table 2-13 contains the bit descriptions.

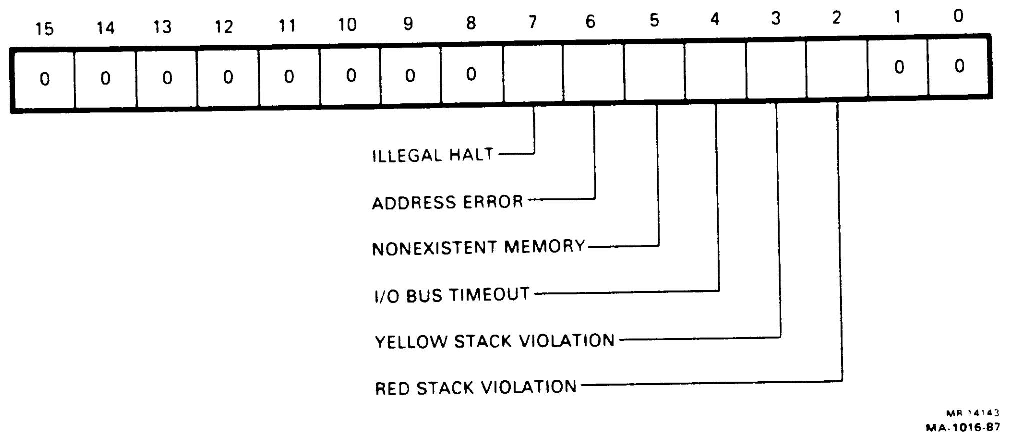

Figure 2-19 CPU Error Register Format

Table 2-13 CPU Error Register Bit Descriptions

Bit(s)	Name	Function
7	Illegal HALT	Set when execution of a HALT instruction is attempted in user or supervisor mode.
6	Address Error (RO)	Set when word access to an odd byte address or an instruction fetch from an internal register is attempted.
5	Nonexistent Memory (RO)	Set when a reference to main memory times out.
4	I/O Bus Timeout (RO)	Set when a reference to the I/O page times out.
3	Yellow Stack Violation (RO)	Set on a yellow zone stack overflow trap.
2	Red Stack Violation (RO)	Set on a red zone stack overflow trap.

2.5.4 Configuration and Display Register

The read-only boot and diagnostic configuration register (BCR) reflects the status of the eight edge-mounted switches at the top of the KDJ11-BF module. All of the switches are also routed to connectors on the KDJ11-BF module to allow them to be asserted remotely. Switches 1-8 control register bits <7:0> respectively.

NOTE
All eight switches on the KDJ11-BF module are off. Setting any of these switches is restricted to special applications. See Appendix C for a more detailed description.

Data bits <2:0> (switches 6-8) are also connected directly to the three baud rate select lines of the console SLU on the KDJ11-BF module. These three bits control the baud rate of the console and override the console serial line baud rate switch. Switches 6-8 are normally off to allow the baud rate select switch on the console serial line board (rear of box or cabinet) to select the baud rate. This eliminates the need to gain access to the CPU module to select the baud rate.

Switches 6-8 would be used to select the baud rate only if the baud rate switch was not present. See Appendix C for additional information.

Data bits <7:3> (switches 1-5) are read by the ROM code to define some of the actions to be taken by the ROM code after power-up or restart. See Appendix C for a detailed description of each of these bits.

Figure 2-20 shows the register format. Bits <15:8> are always read as 0s. The value of bits <7:0> depends on the position of the switches on the CPU module and any external switch which might be connected.

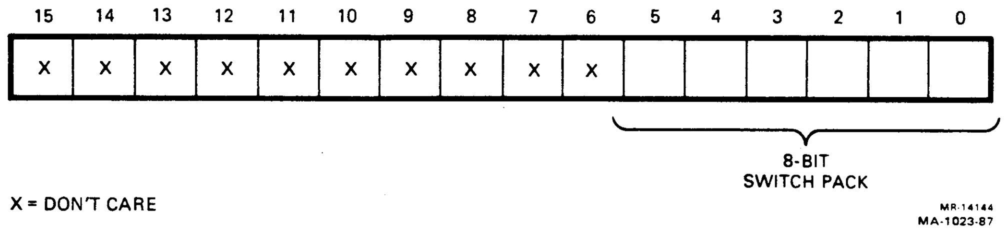

Figure 2-20 Boot and Diagnostic Configuration Register Format

The write-only boot and diagnostic display register (BDR), at address 17777524, allows the boot diagnostic programs to light the front panel start-up test LED display and the LEDs on the KDJ11-B module. These display bits are also available on an external connector. Bits <05:00> are cleared on power-up (all LEDs on) by the negation of DCOK. Figure 2-21 shows the register format and Table 2-14 contains the bit descriptions.

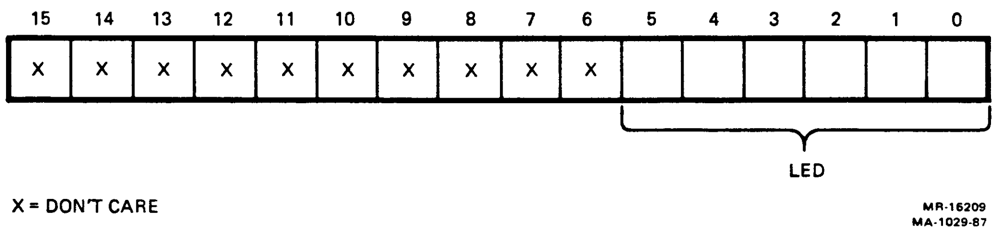

Figure 2-21 Display Register

Table 2-14 Display Register Bit Descriptions

Bit(s)	Name	Function
15:06	-	Unused
05:00	LED 5-0	These bits enable the boot and diagnostic programs to light the LEDs located at the top of the CPU module. Clearing any of these bits lights the corresponding LED.

2.5.5 Maintenance Register

The J-11 microcode addresses the maintenance register at address 17777750 and reads BPOK H, the power-up option code, and the halt/trap option bit. Other bits in the maintenance register, not used by J-11 microcode, contain information on the module type and system parameters useful to operating system and diagnostic software. Figure 2-22 shows the register format and Table 2-15 contains the bit descriptions.

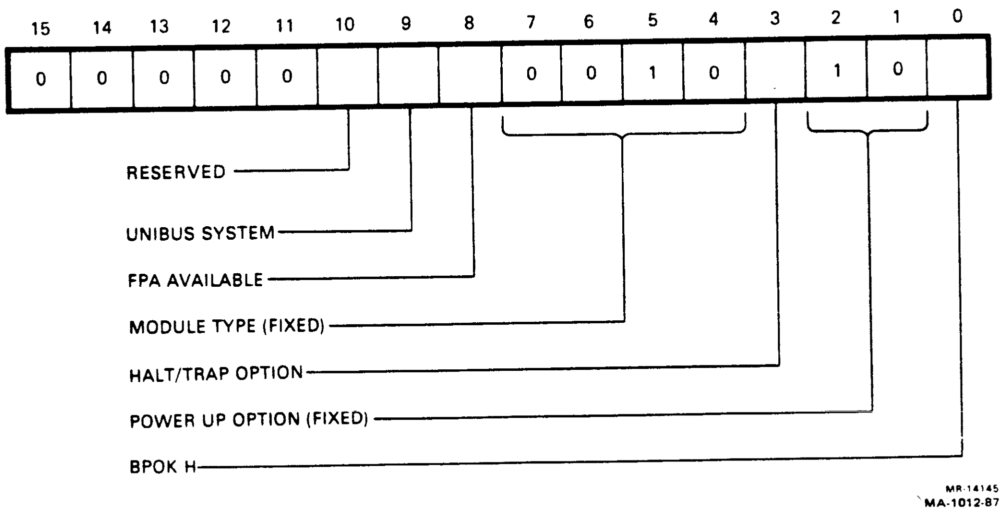

Figure 2-22 Maintenance Register Format

The power-up option code is hard wired for standard bootstrap operation (code 2). The PSW is set to 340, and the processor begins program execution at address 173000. The boot and diagnostic code, which starts at that location, configures the KDJ11-B. The code runs standalone diagnostics before acting on the user-specified power-up option stored as part of the EEPROM configuration data.

Table 2-15 Maintenance Register Bit Descriptions

Bit(s)	Name	Function
15:11	-	Unused. Reserved for future expansion. Read as 0s.
10	-	Unused. Reserved for future use.
09	UNIBUS System (RO)	This bit reflects the status of the externally applied UNIBUS adapter line. A 1 indicates that the system includes a UNIBUS adapter.
08	FPA Available (RO)	When set, this bit indicates that the FPA is available for use.
07:04	Module Type	This 4-bit code is hard-wired as a 2, indicating a KDJ11-BF module.
03	Halt/Trap (R/W)	This read/write bit determines the response of a processor to a kernel mode halt instruction. Setting the bit selects the trap option, causing the CPU to trap to location 4. Clearing the bit selects the halt option, causing the CPU to halt and enter ODT. This bit is cleared by the negation of DCOK and is set by the boot and diagnostic ROM code if the trap option is selected by a bit in the configuration RAM. The trap option is not intended for normal use and is reserved for controller applications.
02:01	Power-Up Code	This 2-bit code is hard-wired as a 2. At power-up, the processor sets the PC to 173000 and sets the PSW to 370. It then starts program execution at location 173000, which is the starting location for the KDJ11-BF boot and diagnostic ROM program. These programs test the KDJ11-BF module and then implement the user-selected power-up option specified in the configuration data.
00	BPOK H	This bit is set (1) if the PMI bus signal BPOK H is asserted, indicating that ac power is acceptable.

2.5.6 Boot and Diagnostic Controller Status Register

The boot and diagnostic controller status register (BCSR), at address 17777520, is both word and byte addressable. Figure 2-23 illustrates the register format and Table 2-16 contains the bit descriptions. The BCSR allows the boot and diagnostic ROM programs to test battery backup status, to set parameters for the processor master grant (PMG) counter and for the line clock, to enable the console halt-on break feature, and to enter or exit from standalone mode.

The BCSR also allows these programs to selectively disable the response of the boot and diagnostic ROMs at addresses 17765000-17765776 and/or at addresses 17773000-17773776 and to control read/write access to the EEPROM memory.

Programs that access the I/O page can use the BCSR to alter PMG and line clock parameters, to enable or disable the halt-on-break feature, and to control access to the ROM and EEPROM memories.

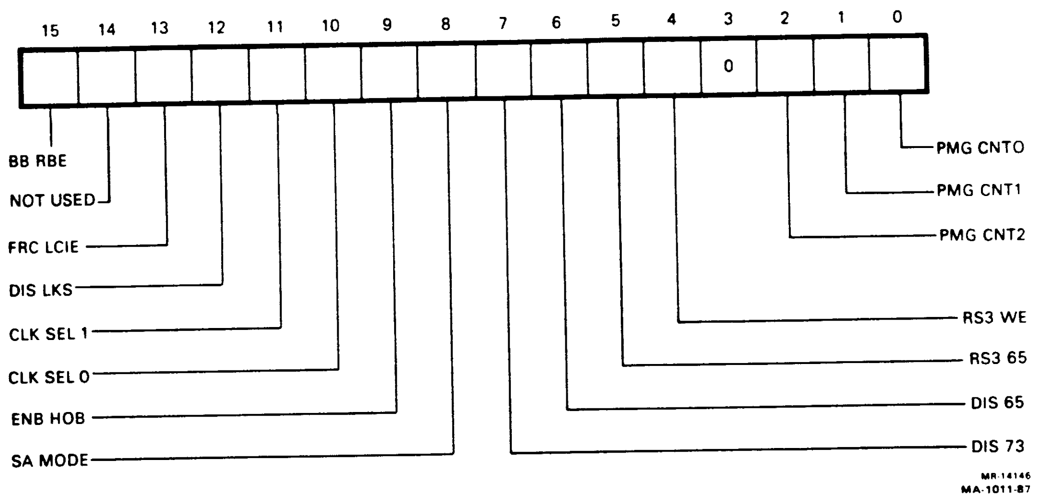

Figure 2-23 Boot and Diagnostic Controller Register Format

Table 2-16 Boot and Diagnostic Controller Register Bit Descriptions

Bit(s)	Name	Function
15	BB RBE	Battery Backup Reboot Enable. When set, this bit indicates that battery backup failed to maintain voltages to the memory system during the previous power failure. When this bit is clear, it indicates that the system does not feature battery backup, or that battery backup maintained voltages during the previous power failure. This signal is received from backplane pin BH1 and latched when DC OK is asserted.
14	-	Unused. Could be a 1 or a 0.
13	FRC LCIE	Force Line Clock Interrupt Enable. If this bit is set, assertion of the signal selected by BCSR <11> and <10> (clock select bits <1> and <0>) will unconditionally request interrupts. If FRC LCIE is clear, assertion of the selected signal will request interrupts only if the line clock status register bit <6> (LCIE) is set under program control. FRC LCIE is cleared by the negation of DCOK.
12	DIS LKS	Line Clock Status Register Disable. If this bit is set, the line clock status register (LKS) is disabled. If this bit is clear, LKS is enabled and responds to bus address 17777546. LKS DIS is cleared by the negation of DCOK.

Table 2-16 (Cont.) Boot and Diagnostic Controller Register Bit Descriptions

Bit(s)	Name	Function
11 10	CLK SEL1 CLK SEL0	Clock Select Bits 1 and 0. These two bits select the source of the line clock interrupt request: **CLK SEL1** / **CLK SEL0** / **Source of Interrupt** 0 / 0 / External LTC Lines 0 / 1 / On-board 50 Hz 1 / 0 / On-board 60 Hz 1 / 1 / On-board 800 Hz Both bits are cleared by the negation of DCOK.
09	ENB HOB (R/W)	Enable Halt on Break. When this bit is set, the console serial line unit halt-on-break feature is enabled. When this bit is clear, the feature is disabled. ENB HOB is cleared by the negation of DCOK.
08	SA MODE (R/W)	Stand-Alone Mode. When this bit is set, the KDJ11-B operates in standalone mode, using its cache as main memory. External memory and peripherals are all disabled. When SA MODE is clear, standalone mode is turned off, enabling external memory and peripherals. SA MODE is set by the negation of DCOK.
07	DIS 73 (R/W)	Disable 17773000. When this bit is set, response of the 16-bit ROM memory to addresses 17773000-17773776 is disabled, allowing the operation of an external ROM that uses those addresses. When DIS 73 is clear, the 16-bit ROMs respond to those addresses, using the high byte of the page control register as the most significant address bits. DIS 73 is cleared by the negation of DCOK.
06	DIS 65 (R/W)	Disable 17765000. When this bit is set, response of the boot and diagnostic 16-bit and 8-bit ROM memory to addresses 17765000-17765776 is disabled; this allows the operation of external ROM which uses those addresses. When DIS 65 is clear, the ROM memory selected by BCSR <5> responds to those addresses, using the low byte of the page control register as the most significant address bits. DIS 65 is cleared by the negation of DCOK.
05	RS3 65 (R/W)	ROM Socket 3 at 17765000. This bit selects whether there is a 16-bit ROM in ROM sockets one and two or there is an 8-bit ROM in ROM socket three corresponding to addresses 17765000-17765776 (assuming that BCSR <4> is clear). If RS3 65 is set, the 8-bit ROM is selected. If RS3 65 is clear, the 16-bit ROM is selected. In either case, the low byte of the page control register provides the most significant address bits. RS3 65 is cleared by the negation of DCOK.

Table 2-16 (Cont.) Boot and Diagnostic Controller Register Bit Descriptions

Bit(s)	Name	Function
04	RS3 WE	ROM Socket 3 Write Enable. If BCSR <6> (DIS 65) is clear, and if BCSR <5> and <4> (RS3 65 and RS3 WE) are both set, then the program can write access ROM socket 3 which typically contains an EEPROM. RS3 WE is cleared by power-up and by bus initialization.
03	-	Unused. This bit always reads as 0.
02 01 00	PMG CNT2 PMG CNT1 PMG CNT0	Processor Master Grant Count bits <2>, <1> and <0>. These three bits enable the PMG counter and select the length of time for PMG counter overflow. When enabled, the PMG counter begins counting when the KDJ11-BF must access an I/O page location or external memory. Counter overflow causes the KDJ11-BF to suppress all DMA requests and give the processor bus master status during the next DMA arbitration cycle. When the PMG counter is disabled, the processor is blocked from bus master status as long as DMA requests are pending. All three bits are cleared by the negation of DCOK.

PMG CNT2	PMG CNT1	PMG CNT0	Count Time µs
0	0	0	(Disabled)*
0	0	1	0.4
0	1	0	0.8
0	1	1	1.6
1	0	0	3.2
1	0	1	6.4
1	1	0	12.8
1	1	1	25.6

*The PMG count of 0 (disabled) is not recommended for most typical systems and is reserved for special applications.

2.5.7 Page Control Register

The page control register, at address 17777522, is a read-write register that is both byte and word addressable. Figure 2-24 shows the register format, and Table 2-17 contains the bit descriptions.

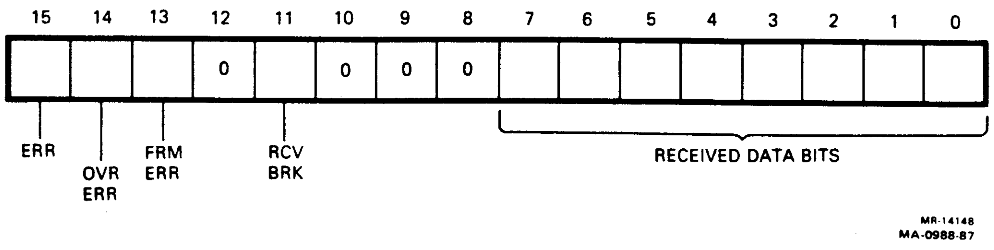

Figure 2-24 Page Control Register Format

Table 2-17 Page Control Register Bit Descriptions

Bit(s)	Name	Function
15	-	Unused. Always read as 0.
14:09	High Byte (R/W)	These six bits provide the most significant ROM address bits when the 16-bit ROM sockets are accessed by bus addresses 17773000-17773776.
08:07	-	Unused. Always read as 0s.
06:01	Low Byte (R/W)	These six bits provide the most significant ROM (or EEPROM) address bits when the 16-bit or the 8-bit ROM (or EEPROM) sockets are accessed by bus addresses 17765000-17765776.
00	-	Unused. Always read as 0.

2.5.8 Line Frequency Clock and Status Register

The line clock provides the system with timing information at fixed intervals determined by the UNIBUS LTC line or by one of the on-board KDJ11-BF frequency signals. The signals are programmed by boot and diagnostic controller status register bits <11> and <10>. Typically, LTC cycles at the ac line frequency, producing intervals of 16.7 ms (60 Hz line) or 20.0 ms (50 Hz line). The three on-board frequencies are 50 Hz, 60 Hz and 800 Hz.

The LKS, at address 17777546, allows line clock interrupts to be enabled and disabled under program control. Alternatively, line clock interrupts can be unconditionally enabled by setting BCSR <13> (FRC LCIE). Program recognition of the clock status register can be disabled by setting BCSR <12> (LKS DIS).

The normal KDJ11-BF configuration is FRC LCIE and LKS DIS both clear. These bits are set up by the boot and diagnostic ROM programs from the KDJ11-BF configuration data. Figure 2-25 shows register format, and Table 2-18 contains the bit descriptions.

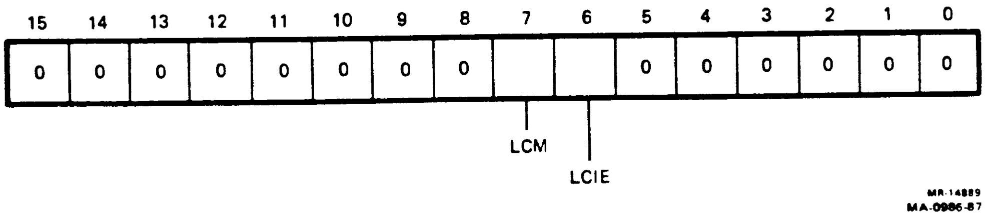

Figure 2-25 Clock Status Register Format

Table 2-18 Clock Status Register Bit Descriptions

Bit(s)	Name	Function
15:08	-	Unused. Always read as 0.
07	LCM (R/W)	Line Clock Monitor. This bit is set by the leading edge of the external BEVENT line (or of one of the three on-board clock frequencies) and by bus initialization. LCM is cleared automatically on processor interrupts acknowledge. It is also cleared by writes to the LKS with bit <7> = 0.
06	LCIE (R/W)	Line Clock Interrupt Enable. This bit, when set, causes the set condition of LCM (LKS <7>) to initiate a program interrupt request at a priority level of 6. When LCIE is clear, line clock interrupts are disabled. LCIE is cleared by power-up and by bus initialization. LCIE is held set INIT. LCIE is held set when BCSR <13> (FRC LCIE) is set.
05:00	-	Unused. Always read as 0s.

2.6 Stack Limit Protection

The KDJ11-BF checks kernel stack references against a fixed limit of 400(8). If the virtual address of the stack reference is less than 400(8), a yellow stack trap occurs at the end of the current instruction.

A stack trap can only occur in kernel mode and only on a stack reference. A stack reference is defined as a mode 4 or 5 through R6, or a JSR, trap, or interrupt stack push.

In addition, the J-11 checks for kernel stack aborts during interrupt, trap, and abort sequences. If during one of these sequences a kernel stack push causes an abort, the J-11 initiates a red zone stack trap. The J-11 sets CPU error register bit <2>, loads virtual address 4 into the kernel stack pointer (R6), and traps through location 4 in kernel data space. The old PC and PS are saved in kernel data space locations 0 and 2, respectively.

NOTE

The J-11 microprocessor treatment of yellow stack trap is identical to that of the PDP-11/44 system. The PDP-11/70 system includes a stack limit register and a more inclusive definition of stack reference. The J-11 processor's definition of a red stack trap is unique.

2.7 Kernel Protection

In order to protect the kernel operating system against interference, the KDJ11-BF incorporates the following protection mechanisms.

1. In kernel mode, HALT, RESET, and set processor level (SPL) execute as specified. In supervisor or user mode, HALT causes a trap through location 4, while RESET and SPL are treated as NOPs.
2. In kernel mode, return from interrupt (RTI) and RTT can freely alter PSW <15:11> and PSW <7:5>. In supervisor or user mode, RTI and RTT can only set PSW <15:11> and cannot alter PSW <7:5>.
3. In kernel mode, MTPS can alter PSW <7:5>. In supervisor or user mode, MTPS cannot alter PSW <7:5>.
4. All trap and interrupt vector references are classified as kernel space references, irrespective of the memory management mode at the time of the trap or interrupt.
5. Kernel stack references are checked for stack overflow. Supervisor and user stack references are not checked.

2.8 Trap and Interrupt Service Priorities

In both traps and interrupts, the currently executing program is interrupted. A new program is then executed, the starting address of which is specified by the trap or interrupt vector. The hardware process for traps and interrupts through a vector V is identical:

```
PS --> temp 1                 !save PS, PC in temporaries
PC --> temp 2
0 --> PS <15:14>              !force kernel mode

M[V] --> PC                   !fetch PC from vector, data space
M[V+2] --> PS                 !fetch PS from vector, data space
temp1<15:14> --> PS<13:12>    !set previous mode
SP-2 --> SP                   !selected by new PS
temp1 --> M[SP]               !push old PS on stack, data space
SP-2 --> SP
temp2 --> M[SP]               !push old PC on stack, data space
                              !go execute next instruction
```

NOTE

If an abort occurs during either the vector fetch or the stack push, the PS and PC are restored to their original state before trap sequence execution.

The priority order for traps and interrupts is as follows:

```
red stack trap
address error
memory management violation
timeout/nonexistent memory
parity error
trace (T-bit) trap
yellow stack trap
power fail
floating-point trap
PIRQ 7
interrupt level 7
event (LTC) line time clock
PIRQ 6
interrupt level 6
PIRQ 5
interrupt level 5
PIRQ 4
interrupt level 4
PIRQ 3
PIRQ 2
PIRQ 1
halt line
```

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

2.9 Console Serial Line Unit

The console serial line provides the KDJ11-BF processor with a serial interface for the console terminal. The console serial line is full duplex. It provides an RS-423 EIA interface which is also RS-232C compatible.

This serial line interface is based on the DC319 Digital link asynchronous receiver transmitter (DLART). For additional details, see the *Chipkit Handbook* (EK-01387-92).

The user should make sure that the console serial receive and transmit baud rates are selected at the console SLU panel using the baud rate switch to match the baud rate of the customer-supplied terminal(s).

The switch settings mounted on top of the KDJ11-BF module, although normally in the off position, can be changed to reconfigure a system in the following manner.

These switch settings plus five additional switch settings (switches 07–03) may be read via the boot and diagnostic facility configuration register (BCR). If switch 07 is on, the boot and diagnostic program assumes that the system does not have a console terminal. The program then uses switches 06–00 to select a limited range of KDJ11-BF and system parameters. Setting switch 07 does not disable the console terminal interface which runs at the baud rate reflected by switches 02–00. The setting of these switches, however, is determined by system configuration considerations.

There are four console serial line unit registers: the receiver status register, the receiver data buffer, the transmitter status register, and the transmitter data buffer. Program recognition of these registers cannot be disabled. These registers are described in the following sections.

2.9.1 Receiver Status Register

Figure 2-26 shows the Receiver Status Register (RCSR) format at address 17777560. Table 2-19 contains the bit descriptions.

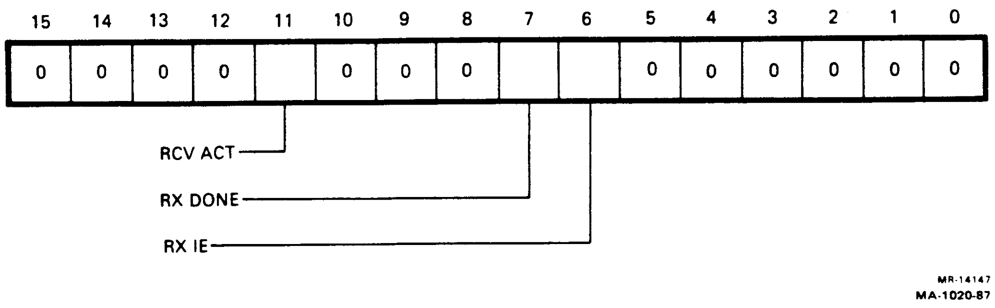

Figure 2-26 Receiver Status Register Format

Table 2-19 Receiver Status Register Bit Descriptions

Bit(s)	Name	Function
15:12	-	Unused. Read as 0s.
11	RCV ACT (RO)	Receiver Active. This bit is set at the center of the start bit of the serial input data. It is cleared at the expected center (per DLART timing) of the stop bit at the end of the serial data. RX DONE is set one bit time after RCV ACT is cleared.
10:08	-	Unused. Read as 0s.
07	RX DONE (RO)	Receiver Done. This bit is set when an entire character has been received and is ready to be read from the RBUF register. This bit is automatically cleared when RBUF is read. It is also cleared by power-up.
06	RX IE (R/W)	Receiver Interrupt Enable. This bit is cleared by power-up and bus initialization. If both RCVR DONE and RCVR INT ENB are set, a program interrupt is requested.
05:00	-	Unused. Read as 0s.

2.9.2 Receiver Data Buffer

Figure 2-27 shows the receiver data buffer register (RBUF) format at address 17777562. Table 2-20 contains the bit descriptions.

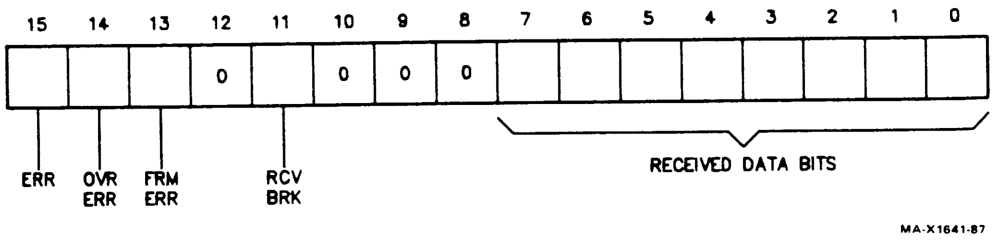

Figure 2-27 Received Data Buffer Register Format

Table 2-20 Received Data Buffer Register Bit Descriptions

Bit(s)	Name	Function
15	ERR (RO)	Error. This bit is set if RBUF <14> or <13> is set. ERR is cleared if these two bits are cleared. This bit cannot generate a program interrupt.
14	OVR ERR (RO)	Overrun Error. This bit is set if a previously received character is not read before being overwritten by the present character.
13	FRM ERR (RO)	Framing Error. This bit is set if the present character has no valid stop bit. This bit is used to detect break.
		NOTE **Error conditions remain present until the next character is received. At that point, the error bits are updated. The error bits are not necessarily cleared by power-up.**
12	-	Unused. This bit always reads as 0.
11	RCV BRK (RO)	Received Break. This bit is set at the end of a received character for the serial data input line. The serial data input line remains in the space condition for all 11-bit time. RCV BRK then remains set until the serial data input returns to the mark condition.
10:08	-	Unused. These bits always read as 0.
07:00	Received Data Bits	These read-only bits contain the last received character.

2.9.3 Transmitter Status Register

Figure 2-28 shows the transmitter status register (XCSR) format at address 17777564. Table 2-21 contains the bit descriptions.

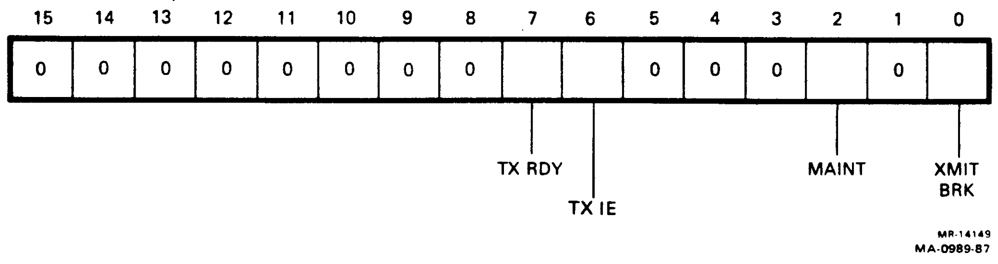

Figure 2-28 Transmitter Status Register Format

Table 2-21 Transmitter Status Register Bit Descriptions

Bit(s)	Name	Function
15:08	-	Unused. Read as 0s.
07	TX RDY (RO)	Transmitter Ready. This bit is cleared when XBUF is loaded. The bit sets when XBUF can receive another character. XMT RDY is set by power-up and by bus initialization.
06	TX IE (R/W)	Transmitter Interrupt Enable. This bit is cleared by power-up and by bus initialization. If both TX RDY and TX IE are set, a program interrupt is requested.
05:03	-	Unused. Read as 0s.
02	MAINT (RO)	Maintenance. This bit is used to facilitate a maintenance self-test. When MAINT is set, the external serial input is disconnected and the serial output is used as the serial input. This bit is cleared by power-up and by bus initialization.
01	-	Unused. Read as 0.
00	XMIT BRK (R/W)	Transmit Break. When this bit is set, the serial output is forced to the space condition. XMIT BRK is cleared by power-up and by bus initialization.

2.9.4 Transmitter Data Buffer Register

Figure 2-29 shows the transmitter data buffer register (XBUF) format at address 17777566. Table 2-22 contains the bit descriptions.

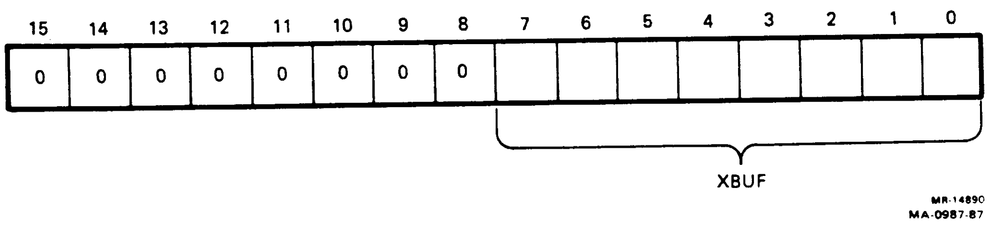

Figure 2-29 Transmitter Data Buffer Register Format

Table 2-22 Transmitter Data Buffer Register Bit Descriptions

Bit(s)	Name	Function
15:08	-	Unused. Always read as 0s.
07:00	XBUF (WO)	These eight bits are used to load the transmitted character.

2.9.5 Break Response

The KDJ11-BF console serial line unit may be configured either to perform a halt operation or to have no response when a break condition is received. A halt operation will cause the processor to halt and enter the octal debugging technique (ODT) microcode. The halt-on-break option is selected via bit 9 of the boot and diagnostic controller status register. During power-up or restart, the boot and diagnostic ROM program will always set bit <9> to a <1>. This enables the halt-on-break condition if the Keylock switch is in the enable position.

The DLART recognizes a break condition at the end of a received character for which the serial data input remained in the space condition for all 11 bit times. The break recognition line remains asserted until the serial data input returns to the mark condition.

2.10 Kernel/Supervisor/User Mode Descriptions

The PDP-11/84 processor family offers three modes of execution: kernel, supervisor, and user. These modes enhance the memory protection scheme and increase the flexibility and functionality of timesharing and multiprogramming environments.

Kernel mode is the most privileged of the three modes and allows execution of any instruction. In an operating system featuring multiprogramming, the ultimate control of the system is implemented in code that executes in kernel mode. Typically, this includes control of physical I/O operations, job scheduling, and resource management.

Memory management mapping and protection allows these executive elements to be protected from inadvertent or malicious tampering by programs executing in the less privileged processor modes. If the I/O page is only mapped in kernel mode, then only the kernel has access to the memory management registers to re-map or modify the protection. This limited access results because the memory management registers themselves exist in the I/O page.

In order for a user program to have sensitive functions performed in its behalf, a request must be made of the executive program. This request is typically in the form of a software trap that vectors the processor into kernel mode. Thus the executive code remains in control and can verify that the function requested is consistent with the operation of the system as a whole.

The supervisor mode is the next most privileged mode. It may be used to provide for the mapping and execution of programs shareable by users but still requiring protection from them. Supervisor mode might include command interpreters, logical I/O processors, or run-time systems.

User mode is the least privileged mode. It prohibits the execution of instructions such as HALT and RESET, as does supervisor mode. A multiprogramming operating system typically restricts execution of user programs to user mode. This restriction prevents a single user from having a negative effect on the system as a whole. The user's virtual address space is set up such that the only areas of memory that can be written are those that belong to that user. Areas shared among users are protected for read-only, execute-only, or for both read and execute access.

2.11 PMG Counter

The PMG counter enables the processor to become PMI master during heavy UNIBUS DMA activity. This allows the processor to limit the hog mode control of the PMI during DMA activity. To change the PMG counter parameters, see Section 3.3, on setup mode. The user can select from seven counter values and one disable. The PMG counter default setting is the disabled mode, with no DMA interruptions from the processor. The least PMG counter timeout, with the exception of the disable mode, is selection 7. Selection 1, the fastest PMG counter timeout, provides the greatest number of PMI interrupts per number of clock cycles. Table 2-23 provides a list of the eight PMG counter selections.

Table 2-23 PMG Counter Timeout List

Switch Position	Count Timeout (μs)
0	(Disabled)*
1	0.4
2	0.8
3	1.6
4	3.2
5	6.4
6	12.8
7	25.6

*The PMG count of 0 (disabled) is not recommended for most typical systems, and is reserved for special applications.

2.12 KTJ11-B Cache Operations

The KTJ11-B DMA cache is used to decrease PMI memory read access time for UNIBUS DMA devices. The cache is divided into four sections. Each section can store up to eight addresses.

Initially, the KTJ11-B cache is flushed (that is, emptied). The first PMI read on an octal boundary from a UNIBUS DMA device causes the KTJ11-B cache to read from memory. It then stores that address and the next seven PMI address locations in section A. If the next PMI read is one of the addresses stored in the cache, a cache hit occurs.

If the next PMI read does not match any cache address (cache miss), the KTJ11-B cache does a memory read cycle for the requested address—if that address is on an octal boundary—and the next seven PMI memory addresses. These eight words are then stored in the next available cache section (that is, B, C, or D).

The KTJ11-B cache also monitors the PMI address lines. If a write into an address occurs, the KTJ11-B cache compares the address with its stored cache addresses. When a match (hit) occurs, the cache address location is invalidated.

2.12.1 KTJ11-B Cache Organization

The KTJ11-B DMA cache contains 32 16-bit data registers. These registers are arranged in four sets (A, B, C, and D) of eight data registers each (000–111). Associated with each set is a valid bit and an 18-bit tag register. The data registers are located in RAM as shown in Table 2-24. The tag registers and valid bits are located in the KTJ11-B gate array. The format is shown in Figure 2-30.

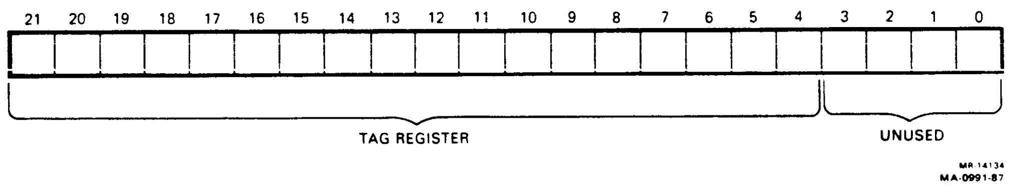

Figure 2-30 Valid Bit and Tag Register Format (One of Four)

Table 2-24 RAM Data Register Locations

RAM Address	Register	RAM Address	Register
00	Set A Register 0	20	Set C Register 0
01	Set A Register 1	21	Set C Register 1
02	Set A Register 2	22	Set C Register 2
03	Set A Register 3	23	Set C Register 3
04	Set A Register 4	24	Set C Register 4
05	Set A Register 5	25	Set C Register 5
06	Set A Register 6	26	Set C Register 6
07	Set A Register 7	27	Set C Register 7
10	Set B Register 0	30	Set D Register 0
11	Set B Register 1	31	Set D Register 1
12	Set B Register 2	32	Set D Register 2
13	Set B Register 3	33	Set D Register 3
14	Set B Register 4	34	Set D Register 4
15	Set B Register 5	35	Set D Register 5
16	Set B Register 6	36	Set D Register 6
17	Set B Register 7	37	Set D Register 7

The KTJ11-B DMA cache interprets the DMA (or CPU) physical address. Figure 2-31 illustrates the register, and Table 2-25 contains the bit descriptions.

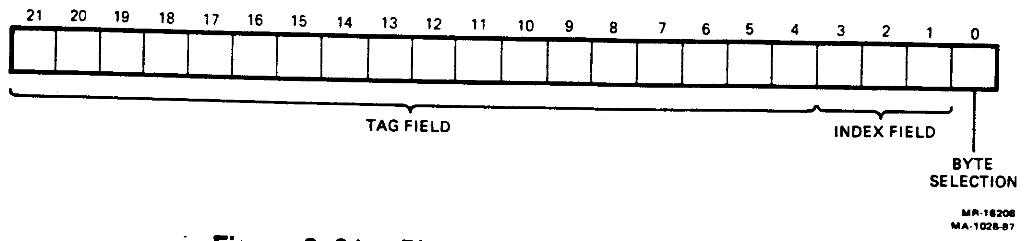

Figure 2-31 Physical Address Interpretation

Table 2-25 UBA Physical Address Interpretation

Bit(s)	Name	Function
21:04	Tag Field	These 18 bits comprise a field that corresponds to the bits in the tag registers.
03:01	Index Field	These three bits indicate if the address is on an even eight-word boundary (index=0) and points to one of eight data registers in a set (A-D).
00	Byte Selection	This bit selects high- or low-byte write operations. This bit has no effect during DMA operations.

2.12.2 DMA Cache Enable/Disable

The KTJ11-B Memory Configuration Register (KMCR) (described in Section 2.13.3) contains both control and diagnostic status bits for the KTJ11-B DMA cache.

When bit <06> of the KMCR is cleared, or when bit <05> of memory management register 3 is cleared (that is, when the UNIBUS map is disabled), then the DMA cache is disabled and initialized to its power-up condition. All four valid bits are cleared. Set A is the next available set, followed by set B, set C, and set D. The 32 data registers and the 4 tag registers are not cleared and contain random information.

When KTJ11-B memory configuration register bit <06> and memory management register 3 bit <05> are both set, the DMA cache is enabled and operates as described in the following sections.

2.12.3 DMA Cache Write Operations

During CPU and DMA write operations, the physical address tag field is checked against the tag register and valid bit for each of the four sets to determine whether a DMA cache hit has occurred. If a cache hit has occured, then the set which caused the hit becomes the next available set and its valid bit is cleared. Otherwise, the DMA cache is not affected (Figure 2-32).

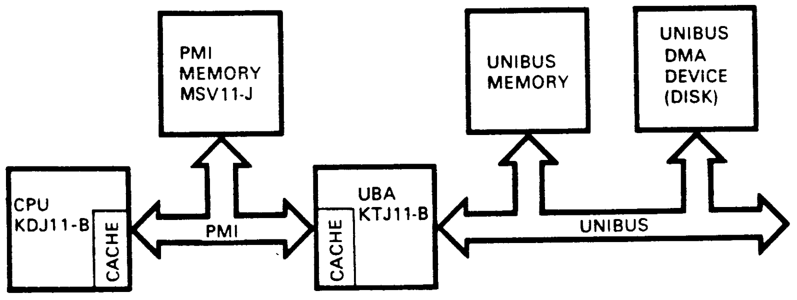

Figure 2-32 PDP-11/84 Cache Diagram

2.12.4 DMA Cache Read Operations

During all DMA reads from PMI memory, the physical address of the tag field is checked against the tag register and valid bit for each of the four sets. This check determines whether a DMA cache hit has occurred. Also, the physical address index field is checked for an even 8-word boundary (index = 0).

If a DMA cache hit has not occurred, and if the index field does not equal zero, then the content of the addressed memory location is gated onto the UNIBUS. The DMA cache is not affected.

If a DMA cache hit has not occurred, and if the index field equals zero, then the following operations take place.

1. The physical address tag field is loaded into the tag register corresponding to the next available set.
2. The content of the addressed PMI memory location is gated onto the UNIBUS.
3. The content of the addressed main memory location and of the seven succeeding memory locations is stored in the data registers of the next available set.
4. If all eight data registers are successfully loaded, the valid bit corresponding to the next available set is set. That set becomes the least available set.
5. If a parity error occurs while one of the memory locations is read, the corresponding valid bit is cleared. The set remains the next available set.

NOTE

The KTJ11-B cache contains no parity error indication. Since the offending memory location is not stored in the DMA cache, the DMA device receives any parity error indications if and when it specifically accesses that memory location.

If a DMA cache hit occurs, then the following operations take place.

1. The set (A–D) whose tag register caused the hit, along with the physical address index field, selects a data register whose contents is gated onto the UNIBUS.
2. If the index field equals 7 (the last data register in the set has been read), the corresponding valid bit is cleared, and that data register set becomes the next available set.
3. If the index field does not equal 7, the valid bit remains set, but the data register set becomes the least available set.

2.13 UNIBUS Mapping

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

UNIBUS address space is 256 Kbyte; the top 8 Kbyte addresses of this space always reference the I/O page. The lower 248 Kbyte of UNIBUS address space can be used by the UNIBUS map to reference physical memory (Figure 2-33).

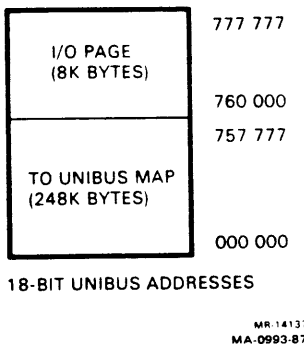

Figure 2-33 UNIBUS Address Space

The UNIBUS map can be programmed, via Memory Management Register 3, (MMR3) bit <05>, to run with relocation enabled or relocation disabled.

If relocation is disabled (MMR3 bit <05> = 0), the UNIBUS map appends four leading zeros (address bits <21:18>) to the UNIBUS address. This produces the 22-bit memory address. Memory address bits <17:00> are identical to UNIBUS address bits <17:00>. If memory address bits <17:13> are all ones, the BBS7 L signal is asserted, selecting the I/O page.

If relocation is enabled (MMR3 bit <05> = 1), the UNIBUS map decodes UNIBUS address bits <17:13> to select one of 31 mapping register pairs (corresponding to octal codes 00 through 36). The content of the selected mapping register pair is added to UNIBUS address bits <12:00> to produce the memory address. If UNIBUS address bits <17:13> are all 1s (octal code 37), the I/O page is selected. The BBS7 L signal to the memory is asserted, memory address bits <17:00> are identical to UNIBUS address bits <17:00> and memory address bits <21:18> are not asserted.

2.13.1 UNIBUS Mapping Registers

The UNIBUS map contains 32 mapping register pairs. Of these pairs, only 31 are actually used for address relocation (Figures 2-34 and 2-35, and Table 2-26). The mapping register pairs may be accessed directly or indirectly.

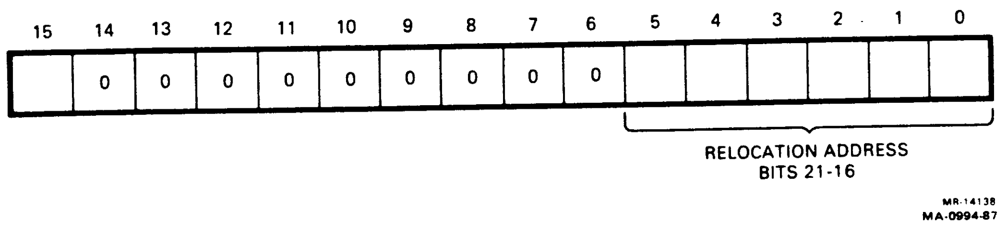

Figure 2-34 High-Address Register Format

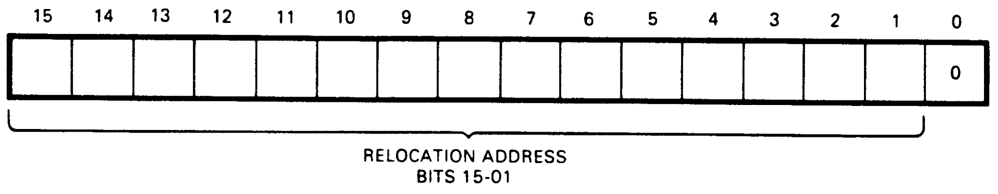

Figure 2–35 Low-Address Register Format

1. Direct Access—The mapping registers are accessed individually through their I/O page addresses. Each mapping register pair consists of a high-address register which contains relocation address bits <21:16>, and a low-address register which contains relocation address bits <15:01>.
2. Indirect Access—When UNIBUS map relocation is enabled, UNIBUS address bits <17:13> select the appropriate mapping register pair to be used in relocating the 18-bit UNIBUS address.

Table 2–26 UNIBUS Map Register Pairs

Register Pair No.	Low-Register	High-Register	UNIBUS Addresses Mapped Via Register Pair
0	17 770 200	17 770 202	000 000–017 777
1	17 770 204	17 770 206	020 000–037 777
2	17 770 210	17 770 212	040 000–057 777
3	17 770 214	17 770 216	060 000–077 777
4	17 770 220	17 770 222	100 000–117 777
5	17 770 224	17 770 226	120 000–137 777
6	17 770 230	17 770 232	140 000–157 777
7	17 770 234	17 770 236	160 000–177 777
10	17 770 240	17 770 242	200 000–217 777
11	17 770 244	17 770 246	220 000–237 777
12	17 770 250	17 770 252	240 000–257 777
13	17 770 254	17 770 256	260 000–277 777
14	17 770 260	17 770 262	300 000–317 777

Table 2-26 (Cont.) UNIBUS Map Register Pairs

Register Pair No.	Low-Register	High-Register	UNIBUS Addresses Mapped Via Register Pair
15	17 770 264	17 770 266	320 000—337 777
16	17 770 270	17 770 272	340 000—357 777
17	17 770 274	17 770 276	360 000—377 777
20	17 770 300	17 770 302	400 000—417 777
21	17 770 304	17 770 306	420 000—437 777
22	17 770 310	17 770 312	440 000—457 777
23	17 770 314	17 770 316	460 000—477 777
24	17 770 320	17 770 322	500 000—517 777
25	17 770 324	17 770 326	520 000—537 777
26	17 770 330	17 770 332	540 000—557 777
27	17 770 334	17 770 336	560 000—577 777
30	17 770 340	17 770 342	600 000—617 777
31	17 770 344	17 770 346	620 000—637 777
32	17 770 350	17 770 352	640 000—657 777
33	17 770 354	17 770 356	660 000—677 777
34	17 770 360	17 770 362	700 000—717 777
35	17 770 364	17 770 366	720 000—737 777
36	17 770 370	17 770 372	740 000—757 777
37*	17 770 374	17 770 376	I/O Page (No Relocation)

* Can be read or written into, but not used for mapping.

2.13.2 Optional UNIBUS Memory

The ability to install UNIBUS memory instead of, or in addition to, PMI memory has been preserved. However, it differs somewhat from previous implementations. UNIBUS address space is assigned to UNIBUS memory in 8 Kbyte segments. Assignment starts with the segment below the I/O page and proceeds downward Table 2-27.

Table 2-27 Register Selection of UNIBUS Memory

04	03	02	01	00	UNIBUS Memory Size (Kbytes)	UNIBUS Memory Address Range*
0	0	0	0	0	0	
0	0	0	0	1	8	XX 740 000—XX 757 777
0	0	0	1	0	16	XX 720 000—XX 757 777
0	0	0	1	1	24	XX 700 000—XX 757 777
0	0	1	0	0	32	XX 660 000—XX 757 777
0	0	1	0	1	40	XX 640 000—XX 757 777
0	0	1	1	0	48	XX 620 000—XX 757 777
0	0	1	1	1	56	XX 600 000—XX 757 777
0	1	0	0	0	64	XX 560 000—XX 757 777
0	1	0	0	1	72	XX 540 000—XX 757 777
0	1	0	1	0	80	XX 520 000—XX 757 777
0	1	0	1	1	88	XX 500 000—XX 757 777
0	1	1	0	0	96	XX 460 000—XX 757 777
0	1	1	0	1	104	XX 440 000—XX 757 777
0	1	1	1	0	112	XX 420 000—XX 757 777
0	1	1	1	1	120	XX 400 000—XX 757 777
1	0	0	0	0	128	XX 360 000—XX 757 777
1	0	0	0	1	136	XX 340 000—XX 757 777
1	0	0	1	0	144	XX 320 000—XX 757 777
1	0	0	1	1	152	XX 300 000—XX 757 777
1	0	1	0	0	160	XX 260 000—XX 757 777
1	0	1	0	1	168	XX 240 000—XX 757 777

*XX = 17 for KMCR <05> = 0 (22-bit mode).
XX = 00 for KMCR <05> = 1 (18-bit mode).

Table 2-27 (Cont.) Register Selection of UNIBUS Memory

04	03	02	01	00	UNIBUS Memory Size (Kbytes)	UNIBUS Memory Address Range*
1	0	1	1	0	176	XX 220 000–XX 757 777
1	0	1	1	1	184	XX 200 000–XX 757 777
1	1	0	0	0	192	XX 160 000–XX 757 777
1	1	0	0	1	200	XX 140 000–XX 757 777
1	1	0	1	0	208	XX 120 000–XX 757 777
1	1	0	1	1	216	XX 100 000–XX 757 777
1	1	1	0	0	224	XX 060 000–XX 757 777
1	1	1	0	1	232	XX 040 000–XX 757 777
1	1	1	1	0	240	XX 020 000–XX 757 777
1	1	1	1	1	248	XX 000 000–XX 757 777

*XX = 17 for KMCR <05> = 0 (22-bit mode).
XX = 00 for KMCR <05> = 1 (18-bit mode).

The UNIBUS address segments assigned to UNIBUS memory cannot be used to access PMI memory via the I/O map. Whenever the CPU accesses UNIBUS memory, the KTJ11-B will disable PMI memory by asserting the PMI UBMEM signal. The KDJ11-B CPU module does not cache UNIBUS memory because it disables its cache when UBMEM is asserted.

NOTE

PDP-11/84 supports *only* 18-bit UNIBUS memories.

The UNIBUS address range of each UNIBUS memory module is determined by jumpers or switches on that module. The KTJ11-B memory configuration register (KMCR), described in Section 2.13.3, must accurately reflect the placement of UNIBUS memory within the system.

The KMCR register is cleared by the assertion of DC LO and is loaded by the KDJ11-B boot and diagnostic programs as specified by the KDJ11-B EEPROM configuration data.

KMCR bits <04:00> specify the number of 8 Kbyte address segments, from 0 to 31, assigned to UNIBUS Memory. KMCR <05> specifies the location of the UNIBUS memory from the viewpoint of the CPU, see Table 2-27. If KMCR <05> is clear (22-bit mode), the top UNIBUS memory location is 17757776. If KMCR <05> is set (18-bit mode), the top UNIBUS memory location is 757776.

If the system has no UNIBUS memory, then KMCR <05:00> must all be cleared. In this configuration:

- PMI memory, as seen by the CPU, resides in contiguous locations from 00000000 up to as high as 17757777.
- All UNIBUS DMA devices access PMI memory through the UNIBUS map.

If the system contains UNIBUS memory only, then KMCR bits <05:00> must all be set. In this configuration:

- UNIBUS memory, as seen by the CPU, resides in contiguous locations from address 0 up to as high as 00757777.
- UNIBUS memory, as seen by all UNIBUS DMA devices, resides in contiguous locations from address 0 up to as high as address 757777.
- The UNIBUS map register pairs are still accessible as read-write registers, but the UNIBUS map is disabled and does not respond to the UNIBUS address range 0 through 757777.

If the system contains both PMI memory and UNIBUS memory, and if KMCR <05> is clear (22-bit mode), then:

- UNIBUS memory as seen by the CPU falls within the 8 Kbyte segments assigned to UNIBUS memory by KMCR <04:00>. UNIBUS memory addresses are assigned downward starting with 8 Kbyte segment 17740000-17757777. See Table 2-27 for the list of the UNIBUS to address space allocated by the various KMCR bit codes.
- PMI memory as seen by the CPU resides in contiguous locations from address 00000000 up to as high as the last address not assigned to UNIBUS memory. The KTJ11-B specifically disables the response of PMI memory residing in locations assigned to UNIBUS memory by asserting the PMI UNIBUS memory line (PUBMEM).
- The UNIBUS DMA devices access PMI memory through the section of the UNIBUS map address space that has not been assigned to UNIBUS memory. Each 8 Kbyte segment assigned to UNIBUS memory disables its corresponding UNIBUS map register pair. Disabled pairs are still accessible as read-write registers, but the UNIBUS map does not respond to their assigned UNIBUS address space.
- The UNIBUS DMA devices access UNIBUS memory directly with an 18-bit address. DMA address XXXXXX accesses the same UNIBUS memory location accessed by CPU address 17XXXXXX.

NOTE

The KTJ11-B places no limit on the number of register pairs that may be disabled by KMCR <04:00>. However, typical system software requires a minimum of five or six register pairs to allow DMA devices to access PMI memory with a moderate degree of efficiency.

If the system contains both PMI memory and UNIBUS memory, and if KMCR <05> is set (18-bit mode), then:

- UNIBUS memory as seen by the CPU falls within the 8 Kbyte segments assigned to UNIBUS memory by KMCR <04:00>. UNIBUS memory addresses are assigned downward starting with 8 Kbyte segment 740000–757777. See Table 2-27, which lists the UNIBUS address space allocated by the various KMCR bit codes.
- PMI memory as seen by the CPU resides in contiguous locations from address 000000 up to as high as the last address not assigned to UNIBUS memory. The KTJ11-B disables the response of PMI memory residing in locations assigned to UNIBUS memory by asserting the PMI UNIBUS memory line (PUBMEM).
- PMI memory as seen by the UNIBUS DMA devices resides in contiguous locations from address 000000 up to as high as the last address not assigned to UNIBUS memory. Because this is an 18-bit system, system software does not enable the UNIBUS map.
- The UNIBUS DMA devices access UNIBUS memory directly, with the same 18-bit addresses used by the CPU.

2.13.3 Memory Configuration Register

The KTJ11-B memory configuration register (KMCR), at address 17777734, allows the KDJ11-B boot and diagnostic programs to configure the KTJ11-B for the distribution of UNIBUS and main memory within the system. Additional KMCR bits allow the DMA cache to be enabled and disabled, provide diagnostic status of the read buffer, and provide information on the system reboot status. Figure 2-36 shows the register format and Table 2-28 contains the bit descriptions.

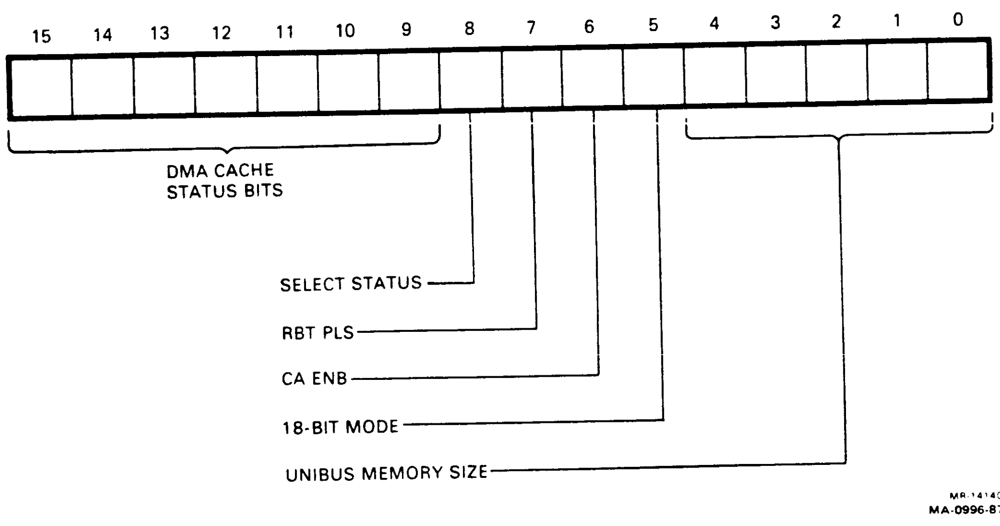

Figure 2-36 Memory Configuration Register (KMCR)

Table 2-28 Memory Configuration Register Bit Descriptions

Bit(s)	Name	Function
15:09	DMA Cache Status Bits (RO)	These seven bits reflect the status of the DMA cache. KMCR <15> is DMA cache hit. The content of KMCR <14:09> depends upon the of the KMCR <08> (status select).
08	Status Select (R/W)	This bit selects the content of KMCR <15:09>.
07	Reboot Pulse (RBT PLS) (RO)	This bit is set by the front panel reboot pulse which also generates a KTJ11-B power-down/power-up cycle. RBT PLS is not cleared by the assertion of DC LO during the KTJ11-B power-down/power-up cycle initiated by the front panel reboot pulse; but it is cleared by any other DC LO assertion.
06	Cache Enable (CA ENB) (R/W)	This bit, when set, enables the DMA cache. When CA ENB is clear, the DMA cache is disabled. CA ENB is cleared by the assertion of DC LO.
05	18-Bit Mode (R/W)	When this bit is set, the CPU can access UNIBUS memory only when address bits <21:18> = 00. When this bit is clear, UNIBUS memory can be accessed if address bits <21:18> = 17. This bit is cleared by the assertion of DC LO. Write access to this bit is disabled when DCSR <08> (diagnostic mode) is clear.
04:00	UNIBUS Memory Size	If the system contains main memory only (no UNIBUS memory), these five bits, as well as KMCR <05>, must be cleared. If the system contains UNIBUS memory only (no main memory), then KMCR <05:00> must be set. If the system contains both main memory and UNIBUS memory, KMCR <04:00> indicate the number of 8 Kbyte address segments assigned to UNIBUS memory. As described in Section 2.4, UNIBUS memory is assigned downward, starting with the segment below the I/O page. These bits are cleared by assertion of DC LO. Write access to these bits is disabled when DCSR <08> (diagnostic mode) is clear.

If KMCR <8> (status select = 0), then KMCR <15:08> contain the DMA cache hit bit, the 2-bit most-recently-used set code, and the four-valid bits. Figure 2-37 shows the field format, and Table 2-29 contains the bit descriptions.

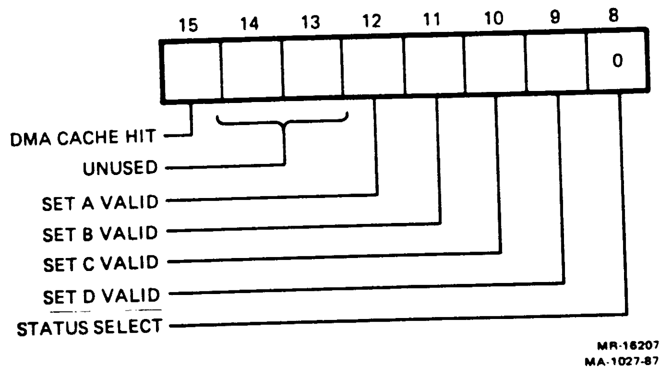

Figure 2-37 Status Select = 0 Field Format

Table 2-29 Status Select = 0 Field Description

Bit(s)	Name	Function
15	DMA Cache Hit	This bit is updated during all writes to main memory and all reads from main memory. It is set if a cache hit is detected and cleared if a cache miss is detected. This bit is cleared when KMCR <6> is clear.
14-13	-	Unused. Always read as 0.
12	Set A Valid	Reflects the current status of the valid bit corresponding to set A. The bit is cleared when KMCR <6> is clear.
11	Set B Valid	Reflects the current status of the valid bit corresponding to set B. The bit is cleared when KMCR <6> is clear.
10	Set C Valid	Reflects the current status of the valid bit corresponding to set C. The bit is cleared when KMCR <6> is clear.
09	Set D Valid	Reflects the current status of the valid bit corresponding to set D. The bit is cleared when KMCR <6> is clear.

If KMCR <8> = 1, then KMCR <15:08> contain the DMA cache hit bit as well as the six bits which determine the relative availability of the four sets (that is, A, B, C, D).

When KMCR <6> (CA ENB) is clear, the DMA cache is disabled and the six set-availability bits are set. Set A is the next-available set, followed by set B, set C, and set D.

When KMCR <6> is set, the DMA cache is enabled. If a DMA cache hit is detected for one of the sets during a CPU or DMA write to memory, then that set becomes the next-available set.

If a DMA cache hit is detected for one of the sets during a DMA read from main memory, then that set becomes the least-available set. Similarly, if a set's data registers are successfully loaded following a DMA read cache miss, then that set becomes the least-available set.

Figure 2-38 shows the field format and Table 2-30 contains the bit descriptions.

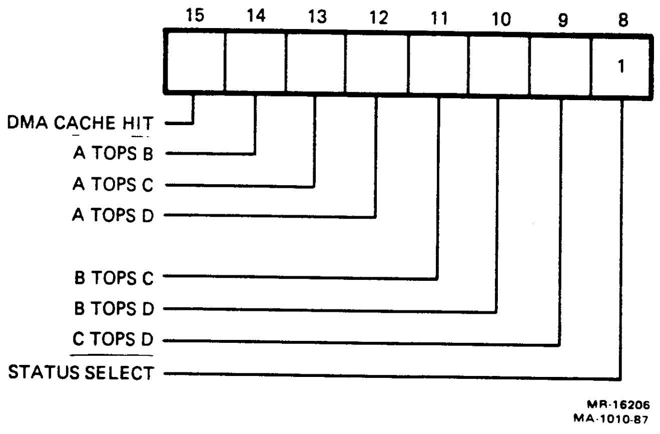

Figure 2-38 Status Select = 1 Field Format

Table 2-30 Status Select = 1 Field Description

Bit(s)	Name	Function
15	DMA Cache Hit	This bit is updated during all writes to main memory and all DMA reads from main memory. It is set if a cache hit is detected and cleared if a cache miss is detected. The bit is cleared when KMCR <6> is clear.
14	A Tops B (ATPSB)	If ATPSB is set, set A is more available than set B. If ATPSB is clear, set B is more available than set A. ATPSB is set when KMCR <6> is clear, when set A becomes the next-available set, and when set B becomes the least-available set. ATPSB is cleared when set B becomes the next-available set, and when set A becomes the least-available set.
13	A Tops C (ATPSC)	If ATPSC is set, set A is more available than set C. If ATPSC is clear, set C is more available than set A. ATPSC is set when KMCR <6> is clear, when set A becomes the next-available set, and when set C becomes the least-available set. ATPSC is cleared when set C becomes the next-available set, and when set A becomes the least-available set.
12	A Tops D (ATPSD)	If ATPSD is set, set A is more available than set D. If ATPSD is clear, set D is more available than set A. ATPSD is set when KMCR <6> is clear, when set A becomes the next-available set, and when set D becomes the least-available set. ATPSD is cleared when set D becomes the next-available set, and when set A becomes the least-available set.
11	B Tops C (BTPSC)	If BTPSC is set, set B is more available than set C. If BTPSC is clear, set C is more available than set B. BTPSC is set when KMCR <6> is clear, when set B becomes the next-available set, and when set C becomes the least-available set. BTPSC is cleared when set C becomes the next-available set, and when set B becomes the least-available set.

Table 2-30 (Cont.) Status Select = 1 Field Description

Bit(s)	Name	Function
10	B Tops D (BTPSD)	If BTPSD is set, set B is more available than set D. If BTPSD is clear, set D is more available than set B. BTPSD is set when KMCR <6> is clear, when set B becomes the next-available set, and when set D becomes the least-available set. BTPSD is cleared when set D becomes the next-available set, and when set B becomes the least-available set.
09	C Tops D (CTPSD)	If CTPSD is set, set C is more available than set D. If CTPSD is clear, set D is more available than set C. CTPSD is set when KMCR <6> is clear, when set C becomes the next-available set, and when set D becomes the least-available set. CTPSD is cleared when set D becomes the next-available set, and when set C becomes the least-available set.

2.14 KTJ11-B Diagnostic and Configuration Registers

The KTJ11-B diagnostic and configuration registers are used with diagnostic programs to check the KTJ11-B in diagnostic mode with the UNIBUS disabled. The KDJ11-B boot and diagnostic programs also use these registers to enable or disable the KTJ11-B DMA cache and to specify the presence and location of UNIBUS memory.

When operating in diagnostic mode, the KTJ11-B can be programmed to perform diagnostic NPR cycles. These cycles test the KTJ11-B address and data paths along with the UNIBUS map.

2.14.1 Diagnostic Controller Status Register

Diagnostic programs use the diagnostic controller status register (DCSR) at address 17777730 to enter and exit from diagnostic mode, to select the source of the Diagnostic Data Register (DDR), and to perform diagnostic NPR cycles that test the UNIBUS map along with the KTJ11-B address and data paths. Figure 2-39 shows the register format and Table 2-31 contains the bit descriptions.

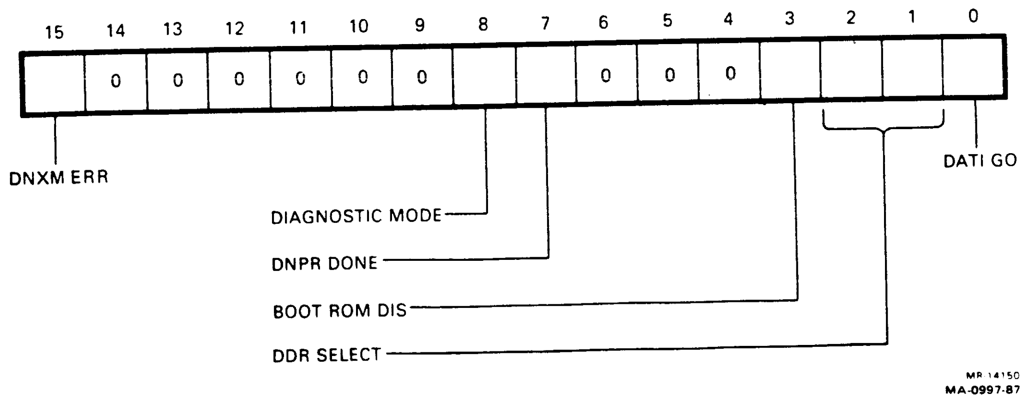

Figure 2-39 Diagnostic Controller Status Register Format

Table 2-31 Diagnostic Controller Status Register Bit Descriptions

Bit(s)	Name	Function
15	DNXM ERR	Diagnostic nonexistent memory error register. This bit is cleared at the start of a diagnostic NPR cycle and set if there is a nonexistent memory timeout during that cycle. DNXM ERR is also cleared when DCSR <08> (diagnostic mode) is cleared.
14:09	-	Unused. These bits always read as 0.
08	Diagnostic Mode (R/W)	When this bit is set, the UNIBUS is disabled and the KTJ11-B is configured for diagnostic mode. When this bit is clear, the UNIBUS is enabled and the KTJ11-B is configured for normal operation. This bit is set by the assertion of DC LO.
07	DNPR Done	This bit is set when there are no diagnostic NPR cycles pending. DNPR Done is cleared by a write to DCSR with a 1 in bit <00>, and by any write to the DDR. DNPR Done is set by bus initialization or by completion of a diagnostic NPR cycle.
06:04	-	Unused. These bits always read as 0.
03	Boot ROM Disable (R/W)	When this bit is set, response of the UBA boot ROM at addresses 177773000-177773776 is disabled, allowing operation of any external ROM which uses those addresses on the UNIBUS. When this bit is cleared, the UBA boot ROM responds to those addresses. This bit is cleared by the assertion of DC LO.
02:01	DDR Select (R/W)	These two bits select the contents of the diagnostic data register during read operations. The DDR select bits are cleared by bus initialization.
00	DATI GO (WO)	Writing a 1 into this bit sets up a diagnostic data-in NPR cycle and clears DCSR bit <07>. The NPR cycle is actually initiated by the next CPU read cycle which accesses the PMI. That cycle provides the address used in the NPR cycle. The data fetched during that cycle is loaded into the DDR.

2.14.2 Diagnostic Data Register

Diagnostic programs use the diagnostic data register at address 17777732, along with the diagnostic controller status register (DCSR). The programs perform diagnostic NPR cycles and monitor the state of various UNIBUS data, address, and control signals.

Diagnostic NPR cycles test the UNIBUS map and many of the KTJ11-B address and data paths. During diagnostic NPR cycles, DCSR <02:01> are set equal to 0, thus selecting the diagnostic NPR register. Following a diagnostic data-in NPR cycle, the diagnostic NPR register contains the transferred data. The data can then be read through the DDR. Diagnostic programs set up a diagnostic data-out NPR cycle by writing the data to be transferred into the DDR. Figure 2-40 shows the register format and Table 2-32 contains the bit descriptions.

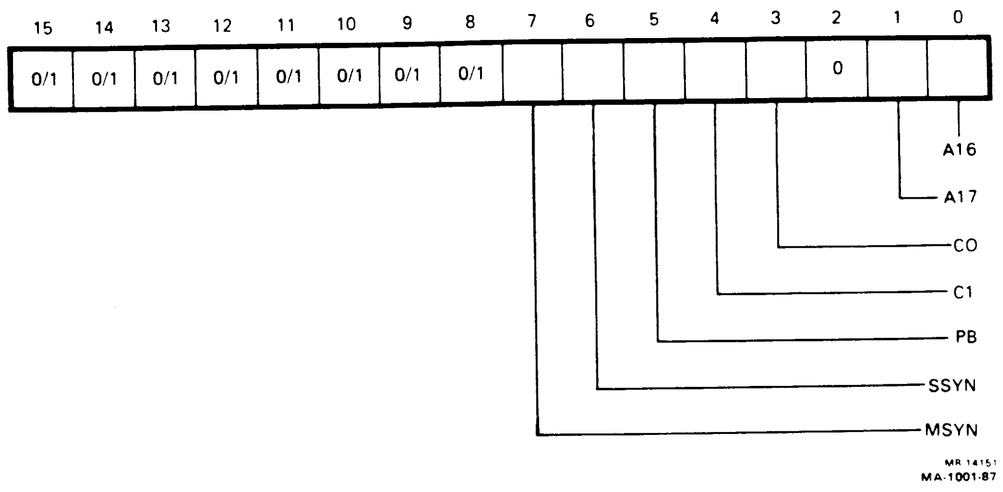

Figure 2-40 Diagnostic Data Register Format

All writes to the DDR access the diagnostic NPR register. The information accessed during read operations from the DDR depends on DCSR <02:01>.

NOTE
Contents of the DDR when select code = 11

Table 2-32 Diagnostic Data Register Content Descriptions

Bit 02	Bit 01	Content of Diagnostic Data Register
0	0	Diagnostic NPR register
0	1	UNIBUS data lines D15-00
1	0	UNIBUS address lines A15-00*
1	1	UNIBUS address lines A17-A16 and various UNIBUS control lines

*Asserted address line A16 during the diagnostic UNIBUS address lines read operation may cause a parity error abort.

2.14.3 Diagnostic DATI NPR Cycles

The execution procedure for diagnostic DATI cycles is as follows.

1. The KTJ11-B must be running in diagnostic mode with DDR select bits (DCSR <02:01>) = 0.
2. The diagnostic program writes a 1 into DCSR bit <00>.

NOTE
At this point, any UNIBUS memory read access or PMI I/O page read or write access results in a bus timeout.

3. The diagnostic program writes the test data pattern into the target memory location. The KTJ11-B latches address bits <A17:00> of this cycle.
4. The KTJ11-B then executes its diagnostic DATI NPR cycle, storing the fetched data in the diagnostic data register. The address used in this cycle is produced by the UNIBUS map, using the latched 18-bit address. The 18-bit address may be used directly (UNIBUS map relocation disabled) or it may be relocated to produce a 22-bit address (UNIBUS map enabled).
5. The diagnostic program verifies that the diagnostic data register contains the correct data.

2.14.4 Diagnostic DATO NPR Cycles

The execution procedure for diagnostic DATO NPR cycles is as follows.

1. The KTJ11-B must be running in diagnostic mode.
2. The diagnostic program loads the data for the NPR cycle into the diagnostic data register. Loading this register primes the KTJ11-B for a diagnostic NPR cycle.

NOTE
At this point, any UNIBUS memory read access or non-PMI I/O page read or write access results in a bus timeout.

3. The KTJ11-B latches address bits <A17:00> from the next KDJ11-B external write to memory address space.
4. The KTJ11-B then executes its diagnostic DATO NPR cycle, using the data stored in the diagnostic data register. The address used in this cycle is produced by the UNIBUS map, using the latched 18-bit address.
5. The diagnostic program verifies that the target memory location contains the correct data. If it wants to check the diagnostic data register, it must do so before performing an external write operation which would alter the contents of that register.

3 Bootstrap and Diagnostic ROM Programming

3.1 Introduction

The CPU contains two read only memories (ROMs). These ROMs store the programs (ROM code) used to test the CPU, UBA, and memory at power-up or restart. They also allow the starting of the user's software on various devices. The data in the ROMs is permanent and cannot be changed by the user.

The CPU also contains an electrically erasable programmable read only memory (EEPROM). The EEPROM stores parameters that the ROM program uses to determine what actions are to be taken at power-up or restart. EEPROM also stores the parameters that determine how various CPU and UBA registers are to be configured.

Parameters in the EEPROM can be changed under control of a program in the ROM called setup mode. (Setup mode does not require the user to remove the CPU or UBA modules.) The EEPROM can also store user bootstrap programs.

The CPU automatically starts the diagnostic ROM program each time the system is powered up or restarted with the Restart/Run/Halt switch on the front panel. The ROM program runs tests selected by parameters in the EEPROM. After testing is complete, parameters in the EEPROM determine what action is to be taken next by the ROM program.

In a typical situation, the ROM program automatically loads and starts a program from the user's disk or tape. This is commonly referred to as booting a program, or automatic boot mode. After the user's software is started, the ROM program is not entered until the system is again powered up or restarted.

In some cases, after testing is complete, the ROM program enters a mode that allows the user to select what action is to be taken next. The user makes selections by entering keyboard commands on the console terminal. This mode is referred to as dialog mode. The parameters in the EEPROM determine the tests to be run, the general mode to be entered after testing is complete, and the final configuration of certain registers on the CPU and UBA modules before the system software is started.

In some cases, the ROM program enters dialog mode regardless of the selections in the EEPROM. This occurs if the user types Ctrl C at the console terminal during testing or the boot sequence, or anytime the forced dialog switch is in the enable position. Generally, this is done by the user to allow changes to be made to the parameters in the EEPROM, or to allow the user to boot a device that was not previously selected by the EEPROM.

The forced dialog switch allows the user to unconditionally override the selections in the EEPROM. Without this override certain modes could not be aborted because the ROM program executes the modes too quickly; the ROM is not able to monitor the console terminal for a Ctrl C typed by the user.

NOTE

All user input is ignored at the console keyboard until the "Testing in progress—Please wait" message is displayed by the ROM code.

The Return key is specified in the examples and text as Return.

The description of the commands for the ROM code assumes that the EEPROMs installed are at Version 8.0 (V8.0). Earlier PDP-11/84 systems contain EEPROMs with V6.0 and V7.0 ROM code. The version of the ROM code is displayed each time setup mode is entered from dialog mode. The version number is displayed at the upper right corner of the printout. It is not necessary to remove the CPU module to determine the ROM code version number. The following list shows the ROM part numbers and version numbers.

Socket Location on CPU (M8190)	Part Number V8.0	Part Number V7.0	Part Number V6.0
E116 (low byte)	23-168E5-00	23-116E5-00	23-077E5-00
E117 (high byte)	23-169E5-00	23-117E5-00	23-078E5-00

Differences between V8.0, V7.0, and V6.0 ROM code are described in Appendix E.

The following examples are messages the ROM program would type or display on the console terminal during power-up or restart of the CPU.

Example 3-1 shows a typical system booting up in automatic boot mode. In this case, the user's software is RT-11 and it is booted from device DU unit 0.

```
Testing in progress - Please wait
Memory Size is 1024 K Bytes
9 Step memory test
Step 1 2 3 4 5 6 7 8 9

Starting automatic boot

Starting system from DU0

RT-11FB (S) V05.0
```

Example 3-1 Automatic Boot Mode

The messages that follow the line "Starting system from DU0" come from the software booted and are not generated by the ROM program. At this point, the ROM program is not executing and all actions are determined by the user's software.

Example 3-2 shows a typical system powering up, running the internal diagnostics, and then entering dialog mode. The ROM program waits for the user to select the action that is to occur next.

```
Testing in progress - Please wait
Memory Size is 1024 K Bytes
9 Step memory test
Step 1 2 3 4 5 6 7 8 9

Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key:
```

Example 3-2 System Power-Up in Dialog Mode

3.2 Dialog Mode Commands

Dialog mode allows the user to:

1. Boot a device
2. List boot programs available to the user
3. Execute ROM-resident tests
4. Provide a map of all memory and I/O page locations
5. Enter setup mode
6. Request help

When dialog mode is entered, the ROM program displays a message at the console terminal (Example 3-3) and waits for the user to select a command.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key:
```

Example 3-3 Dialog Mode Commands

When dialog mode is entered, the user has six commands to choose from. The six commands are listed in the command line for user convenience. The user may obtain a brief description of each command by typing H and pressing Return or by typing a question mark (?) only.

All of the commands may be executed by typing only the first letter of the command and pressing Return. For example, the Map command can be invoked by typing either M or MA or MAP and pressing Return. On input, all lowercase letters are converted to uppercase. Leading spaces and tabs are ignored.

The Delete key deletes the previous character typed. If the terminal type selection in the EEPROM is video, the ROM code erases the previous character on the screen when Delete is pressed. If the terminal type is hardcopy, the ROM code displays slashes (/) to identify all deleted characters.

NOTE
All user inputs in the following examples are in bold type.

The user may at any time delete the entire command line by pressing Ctrl U (Example 3-4). When the user presses Ctrl U, the ROM code deletes the line and displays the prompt.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key:
B DX5 Ctrl U

KDJ11-B >
```

Example 3-4 Ctrl U Executed

When the user presses Ctrl R, the command line is retyped. Ctrl R is normally used when the terminal type is hardcopy; Ctrl R clears command lines (removes slashes) where Delete has been used.

NOTE
To type Ctrl U or Ctrl R, press the Ctrl key while simultaneously pressing the U or R key.

For both Ctrl R and Ctrl U, the ROM code displays a prompt. Neither Ctrl U nor Ctrl R is echoed by the ROM code. Example 3-5 shows how Ctrl R clears the command line without the user retyping all of it. The terminal type selection is hardcopy.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: B DX/X/U1  Ctrl R

KDJ11-B >B DU1
```

Example 3-5 Ctrl R Executed

Input is limited to 16 characters and spaces. There are no cases where any of the commands would need more than 16 characters. If the user types more than 16 characters the ROM code will delete all of the input, display the KDJ11-BF prompt, and wait for input. Example 3-6 shows what happens when more than 16 characters are typed. Typing the seventeenth character is equivalent to pressing Ctrl U.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: 12345678901234567

KDJ11-B >
```

Example 3-6 Ctrl U Equivalent

The ROM code ignores any space or tab typed before a character, or the second tab or space typed in a row without a printable character in between. All tabs are converted to and echoed as spaces.

Note that these rules also apply generally at any time the ROM code is accepting input from the user.

If invalid input is received, an "invalid entry" message is displayed and more input is requested. Example 3-7 shows an invalid entry.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: MP Return

Invalid entry

Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key:
```

Example 3-7 Invalid Entry

3.2.1 Help Command

The Help command displays a brief description of all available commands. It can be executed by either typing H Return, or by typing a question mark (?) only. Dialog mode restarts at the end of this command. Example 3-8 shows the Help command being executed.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: H Return

Command     Description

Help        Type this message
Boot        Load and start a program from a device
List        List boot programs

Setup       Enter Setup mode
Map         Map memory and I/O page
Test        Continuous self test - Type Ctrl C to exit

Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key:
```

Example 3-8 Help Command Display

3.2.2 Boot Command

This command allows a device to be bootstrapped. The command arguments are the device name and the unit number. If the device name is left off, the program prompts the user for it. If the unit number is left off, the program assumes that unit 0 is requested. The unit number ranges from 0 0 to 255_{10}, depending on the device and the boot program. The device name is a one-letter or two-letter mnemonic which describes the device. In most cases, the device name is two letters.

When typing the Boot command, the user may follow either of the following formats:

- B, $\boxed{\text{Return}}$, and then the device name, unit number, and optional switches

  or

- B, a space, the device name, unit number, and optional switches.

Three optional switches may be used with the Boot command:

/A	Request to allow the user to type in a nonstandard CSR address for the controller
/O	The unit number is octal instead of decimal for unit numbers greater than 7
/U	If the boot exists in the base ROM and also on the UBA, override the base ROM boot and use the boot from the UBA board or M9312 module.

When using a switch in the format, the user types the device name, the unit number followed by a slash (/), and the switch reference. When there is more than one switch, only one slash is used.

When the user types the Boot command without an argument, the ROM code prompts the user for additional information with the following message:

```
Enter device name and unit number then press the RETURN key:
```

At this point if the user types a question mark (?), the ROM code lists the boot programs available and then displays the message "enter device name and unit number."

When the user enters a device name, the ROM code searches for the first boot program with the same device name. The ROM code looks for matches in the following order.

- First area to search—EEPROM
- Second area to search—CPU ROM code
- Third area to search—UBA module
- Fourth area to search—M9312 module, if present

The /U switch effectively tells the ROM code not to look for the device name in the EEPROM or the CPU ROM; rather it should go directly to the UBA ROMs or the M9312 module, if present.

Since the EEPROM is always searched first, any boot loaded into the EEPROM with the same device name as a boot in the CPU ROM effectively replaces that boot. This allows a user to replace a CPU ROM boot by loading a boot in the EEPROM with the same name.

Table 3-1 describes how the ROM code interprets user input.

Table 3-1 ROM Code Action

User Input	ROM Code Action
B DL	Boot DL0
B DL1	Boot DL1
B DU8	Boot DU unit 8
B DU10/O	Boot DU unit 8
B DU10 /A Address = 17760400	Boot DU10 with nonstandard CSR address of 17760400
B DU 3/U	Boot DU3 using UBA or M9312 ROM boot instead of CPU ROM code
B DU11/UO	Boot DU unit number 9 using UBA or M9312 ROM boot instead of CPU ROM
B DU 10:	Boot DU10
B DU11/U/O	Invalid format, causes invalid entry error message
B D U 0	Invalid format, no space allowed in the device name DU
BDU0	Invalid format, there must be a space between the Boot command and the device name

If the user types a colon after the unit number, the colon is ignored (for example, B DL1:).

The single letter device name of B implements a method of supporting non-Digital boot devices on the UNIBUS. The letter B causes the ROM code to transfer control to the address location 17773024 of a ROM on the UNIBUS if the address in location 17773024 is not odd.

When the CPU ROM passes control, the CPU ROMs and the UBA ROMs are disabled. R0 will contain a unit number and R1 will contain 0 unless an address is passed by the translation table. If the address in location 17773024 on the UNIBUS is odd, the ROM program displays an invalid-device message. If the UNIBUS device does not respond to all addresses from 17773000 to 17773776, the ROM program again displays an invalid-device message.

The single-letter device name B is used when a module in the system has a switch pack that responds at address 17773024 similar to a M9312 module. The starting address of the program desired is set in the switchpack on the module.

Example 3-9 shows DL2 being booted using the Boot command.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: B DL2 Return

Trying DL2

Starting system from DL2

RT-11FB (S) V05.01

.SET TT QUIE

.R DATIME
Date? [dd-mmm-yy]?
```

Example 3-9 DL2 BOOT

3.2.3 List Command

The List command displays a list of all available boot programs in the CPU ROM, the CPU EEPROM, or any M9312-type ROMs located on the UBA or an M9312 module. The display lists the device name, allowable unit number range, source of the boot program, and a device description. The device name is normally a one- or two-letter mnemonic. The device name must always be a letter(s) (A to Z) as opposed to a numeral(s).

At input, the ROM program converts all lowercase letters to uppercase. The unit number range is the allowable range of unit numbers that is valid for a particular boot program. The range varies from 0 to 255, depending on the device. If the unit number range information is blank, the ROM code assumes the range limit is 0 to 255. The unit number range for M9312-type ROMs is always left blank.

The source lists the actual location of the boot program. The device description is the name of the device to be booted, for example, the mnemonic DL for the RL02.

Dialog mode restarts at the completion of the List command. The mnemonic for each ROM found on either the UBA or the M9312 is checked against a list of mnemonics in the ROM code. If a mnemonic matches, the ROM code displays a description of that device. If no match is found, the description is left blank for that mnemonic. In order for an M9312-type ROM to be listed, it must be in an M9312-type format as described in Sections 3.6.3 through 3.6.6.

Example 3–10 shows a screen display for the List command.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: L [Return]

Device Unit

Name   Numbers    Source      Device type

DU     0-255      CPU ROM     RD51, RD52, RX50, RC25, RA80, RA81, RA60
DL     0-3        CPU ROM     RL01, RL02
DX     0-1        CPU ROM     RX01
DY     0-1        CPU ROM     RX02
DD     0-1        CPU ROM     TU58
DK     0-7        CPU ROM     RK05
MU     0-255      CPU ROM     TK50, TU81

Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key:
```

Example 3–10 List Command

NOTE
Example 3–10 is a model and may not represent the exact screen display.

3.2.4 Setup Command

The Setup command enables the user to list and/or change all parameters in the EEPROM including boot parameters. This command and all of the programmable parameters are discussed in detail in Section 3.3

3.2.5 Map Command

The Map command first displays the CPU crystal frequency. Then it tries to identify all memory in the system and map all locations in the I/O page. Memory is mapped from location 0 to the I/O page in 1,024 byte increments. Memory is not mapped for every location. The routine identifies the size of each memory, the CSR address for each memory if applicable, the CSR type (ECC or parity) and the general bus type.

It is important to note that if two memories share some common addresses or have CSRs with the same address, the Map command will not work properly.

During mapping of memory, if two or more memories are present and they are not contiguous, the ROM code separates their descriptions with a blank line.

After all memory is mapped, the ROM code waits for the user to press [Return] to continue the map. The ROM code then displays all addresses in the I/O page that respond. The I/O page map goes from addresses 17760000 to 17777776. In addition, all addresses that respond that are on the KDJ11-BF or on the KTJ11-B are provided with a short description. There is no description for addresses that respond and are on the external bus, with the exception of memory CSRs.

Dialog mode restarts at completion of the Map command. The ROM code waits for the user to press Return rather than scrolling data forward on video screen terminals. The ROM code always assumes the terminal can display at least 24 lines of 80-column data.

Example 3-11 shows a Map command printout.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: M Return

18.000 MHz
CPU Options: FPA

Memory Map
Starting   Ending        Size in    CSR          CSR     Bus
Address    Address       K Bytes    Address      Type    Type

00000000 - 07777776      2048       17772100     Parity  PMI

Press the RETURN key when ready to continue Return

I/O Page Map
Starting   Ending
Address    Address

17765000 - 17765776      CPU ROM or EEPROM
17770200 - 17770376      UNIBUS Map
17772100                 Memory CSR
17772200 - 17772276      Supervisor I and D PDR/PAR's
17772300 - 17772376      Kernel I and D PDR/PAR's
17772516                 MMR3
17773000 - 17773776      CPU ROM or UBA ROM
17774400 - 17774406
17777520 - 17777524      BCSR, PCR, BCR/BDR
17777546                 Clock CSR
17777560 - 17777566      Console SLU
17777572 - 17777576      MMR0,1,2
17777600 - 17777676      User I and D PDR/PAR's
17777730 - 17777734      DCSR, DDR, KMCR
17777744 - 17777752      MSER, CCR, MREG, Hit/Miss
17777766                 CPU Error
17777772                 PIRQ

Press the RETURN key when ready to continue
```

Example 3-11 Map Command

3.2.6 Test Command

The Test command causes the ROM code to run most of the power-up tests in a continuous loop. The ROM code starts at test 70, runs all applicable tests, and then restarts the loop after test 30 is complete. If an error occurs, the general error routine is entered. The user may exit the test loop by typing Ctrl C at the console. At the time the test loop is exited, the ROM code displays the total number of loops and the total number of errors, if any.

The user may also type a test number after the Test command. If the test is applicable, the ROM code loops on that specific test only until an error occurs or the user types Ctrl C. If the test number selected is not a test that loops, the general test loop is entered and all loopable tests run.

NOTE
Ctrl C is not echoed by the ROM code on the console terminal.

Example 3-12 shows a user entering the Test command. The user types T Return to run all loopable tests. The user aborts the testing sequence after four passes by typing Ctrl C.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: T Return

Continuous self test - Type Ctrl C to exit Ctrl C

Total Passes = 4
Total Errors = 0

Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key:
```

Example 3-12 Test Command

In Example 3-13, the user has requested a loop on only test 60. The user aborts the test loop by typing Ctrl C after 202 passes.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: T 60 Return

Looping on test 60 - Type Ctrl C to exit Ctrl C

Total Passes = 202
Total Errors = 0

Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key:
```

Example 3-13 Loop-On-Test

3.3 Setup Mode Commands

The user enters setup mode by typing S Return in dialog mode. Setup mode allows the user to list or change most of the parameters in the EEPROM. Setup mode also allows changes to any bootstrap programs stored in the EEPROM. Setup mode has 15 commands.

After power-up or restart and the completion of all tests, the ROM code loads the first 105 bytes of the EEPROM into memory, beginning at location 2000. This area in memory is referred to as the setup table. The setup table contains all of the parameters except the EEPROM-resident boot programs.

The EEPROM contains various types of information for the system. The first 105 bytes contain information needed by the ROM code to configure the KDJ11-BF CPU and the KTJ11-B UBA, and to determine the boot device, test selections, and modes. Other information in the EEPROM can be user bootstrap programs and a foreign language file. Setup mode allows changes to the first 105 bytes and to the user bootstrap programs. The foreign language area, if present, cannot be changed in setup mode.

When the user enters setup mode, it displays a list of all commands and provides a short description of each command. Example 3-14 shows setup mode being entered from dialog mode after the user types S Return.

```
Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: S Return

KDJ11-B Setup mode      KDJ11-B ROM V8.0

Command    Description

   1       Exit

   2       List/change parameters in the Setup table
   3       List/change boot translations in the Setup table
   4       List/change the Automatic boot selections in the Setup table
   5       List/change optional user data in the Setup table
   6       List/change the switch boot selections in the Setup table
   7       List boot programs

   8       Initialize the Setup table
   9       Save the Setup table into the EEPROM
  10       Load EEPROM data into the Setup table

  11       Delete an EEPROM boot
  12       Load an EEPROM boot into memory
  13       Edit/create an EEPROM boot
  14       Save boot into the EEPROM
  15       Enter ROM ODT

Type a command then press the RETURN key:
```

Example 3-14 Setup Mode Command Descriptions

The following sections provide a detailed description of each command. To execute a command, the user types the command number and presses Return. At any time, the user may type Ctrl C to return to dialog mode, or Ctrl Z to return to the beginning of setup mode.

NOTE

Never terminate a change of any parameter with Ctrl C or Ctrl Z. If this is done, the change is ignored and lost. Always press Return to terminate a change, and then use Ctrl C or Ctrl Z.

When the user restarts setup mode by typing Ctrl Z, or at the completion of commands 2 through 15, the ROM code displays a short command message instead of the full list of commands. The user may either type in a new command or press Return to list the full command menu. Example 3-15 shows the short command message.

```
KDJ11-B Setup mode
Press the Return key for Help
Type a command then press the Return key:
```

Example 3-15 Short Command Message

3.3.1 Setup Command 1

This is the exit command for setup mode. It returns the user to dialog mode. Dialog mode can also be entered if the user presses Ctrl C.

3.3.2 Setup Command 2

Command 2 prints out the current status of various parameters and allows the user to change them. When setup command 2 is entered, the ROM code displays the current status of all parameters, repeats the first parameter, and waits for user input. The user can press Return to position the program at the parameter to be changed. The user can also go directly to the parameter by typing the letter to the left of the parameter in the first list.

NOTE

After changing any parameters in the setup table, command 9 (Save) should be executed.

To change a parameter, the user types in the new value and presses Return. Pressing Return, Line Feed or period (.) causes the ROM code to proceed to the next parameter. Typing a caret (^) or a minus sign (-) causes the ROM code to proceed to the previous parameter. Any of these characters can be used to change a value.

Example 3-16 shows command 2 being entered. This example also shows the values of the parameters if command 8 is executed in setup mode to initialize the setup table.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 2 [Return]

List/change parameters in the Setup table

A - ANSI Video terminal (1)                     0=No,  1=Yes   = 1
B - Power up  0=Dialog, (1)=Automatic,          2=ODT, 3=24    = 1
C - Restart   0=Dialog, (1)=Automatic,          2=ODT, 3=24    = 1
D - Ignore battery                              0=No,  1=Yes   = 0
E - PMG 0-(7)  1=.4 us, 2=.8, 3=1.6, 4=3.2,...7=25.6  = 7
F - Disable clock CSR                           0=No,  1=Yes   = 0
G - Force clock interrupts                      0=No,  1=Yes   = 0
H - Clock  0=Power supply,  1=50Hz,       2=60Hz, 3=800Hz  = 0
I - Enable ECC test (1)                         0=No,  1=Yes   = 1
J - Disable long memory test                    0=No,  1=Yes   = 0
K - Disable ROM     0=No, 1=Dis 165, 2=Dis 173, 3=Both   = 0
L - Enable trap on Halt                         0=No,  1=Yes   = 0
M - Allow alternate boot block                  0=No,  1=Yes   = 0
N - Disable Setup mode                          0=No,  1=Yes   = 0
O - Disable all testing                         0=No,  1=Yes   = 0
P - Enable UNIBUS memory test (1)               0=No,  1=Yes   = 1
Q - Disable UBA ROM                             0=No,  1=Yes   = 0
R - Enable UBA cache (1)                        0=No,  1=Yes   = 1
S - Enable 18 bit mode                          0=No,  1=Yes   = 0
List/change parameters in the Setup table
Type Ctrl Z to exit or press the RETURN key for No change

ANSI Video terminal (1)          0=No,    1=Yes  = 0      New =
```

Example 3-16 Setup Command 2

NOTE

If 124 Kwords of UNIBUS memory are present, the last two parameters will not be present (enable UBA cache and enable 18-bit mode). When this condition occurs, UBA cache is always disabled and 18-bit mode is forced unconditionally.

The following sections describe each command 2 parameter specified in Example 3-16.

A—ANSI Video Terminal Present

When set to 1, the console terminal is an ANSI video terminal. When 0, the console terminal is hardcopy or non-ANSI compatible. When video terminal is selected, the [Delete] key will erase the previous character on the screen. The ROM code accomplishes this by sending a backspace, space, then backspace to the console terminal. When hardcopy terminal is selected and the [Delete] key is used, the ROM code identifies deleted characters by using the slash (/) character. At power-up, if ANSI video terminal is selected, the ROM code sends an ANSI SCREEN CLEAR and then positions the cursor at line 9 column 1. The video terminal parameter is used only by the ROM code. This information is not used by the operating system.

B—Power-Up Mode and C—Restart Mode

These are two separate parameters. When the ROM code is started, it checks a status bit to determine if the unit is powering up or if the front panel Restart/Run/Halt switch was activated. The ROM code then uses the appropriate mode selected. There are four choices for the power-up mode and the restart mode. The user may define the action taken by the ROM code at power-up or restart to be the same or different.

> **0—Dialog mode.** At completion of the diagnostics, dialog mode is entered. Dialog mode is also entered any time the forced dialog switch is set to the enable position regardless of the mode selected here. (See Section 3.1.)
>
> **1—Automatic mode.** At completion of the diagnostics, the ROM code enters an automatic boot routine that tries to boot a previously selected device. The devices are selected in the EEPROM (see Section 3.3.4). The list of devices can be from one to six devices long. Each device is tried until a successful boot occurs or there are no more devices to try. The default list of devices is A, DL0, MS0, and MU0 in this order. "A" is a special single-letter mnemonic. It causes the ROM code to boot the first disk MSCP device that it can. Removable media devices are tried before fixed media devices.
>
> **2—ODT mode.** At completion of a limited set of tests, the ROM code executes a halt instruction and passes control to J-11 micro ODT (see Section 3.7). If the user types P at this point without changing any registers, the ROM code continues normal testing and then enters dialog mode. This mode is usually used to debug environments. The ROM code will not change any locations in memory before entering ODT mode.
>
> **3—24 mode.** At completion of a limited set of tests, the ROM code loads the PSW with the contents of location 26 and then transfers control to the address located in location 24. This mode is used when nonvolatile memory is present and power-fail recovery is desired. The ROM code does not change any locations in memory before executing mode 24.

D—Ignore Battery

This parameter is used only when the current power-up or restart mode is set to 24 (3). Normally, this parameter is set to 0 meaning that the memory battery-ok signal must be present in order to execute mode 24. If this parameter is set to 1, mode 24 is executed regardless of the status of the battery. At power-up, if the mode selected is mode 24, the ignore-battery parameter is set to 0, and the battery status indicates that voltages were not maintained. The ROM code ignores the power-up selection and uses the restart selection. If the restart selection is also mode 24, the ROM code defaults to dialog mode.

E—PMG Count

This parameter sets the value of the processor mastership count in the BCSR. The range is 0 to 7. When set to 0, the counter is disabled. When set, the count value enables the KDJ11-BF to suppress DMA requests and give the processor bus mastership during the next DMA arbitration cycle after the counter overflows. The processor will only take the bus for one cycle before relinquishing control of the bus to a requesting device. The following table shows the time needed for the counter to overflow for the different values of the PMG count. This parameter is normally set to 7.

Value	Time for Counter to Overflow
0	Disabled*
1	0.4 μs
2	0.8 μs
3	1.6 μs
4	3.2 μs
5	6.4 μs
6	12.8 μs
7	25.6 μs

*The PMG count of 0 (disabled) is not recommended for most typical systems, and is reserved for special applications.

F—Disable Clock CSR

When set to 1, this parameter disables the clock CSR at address 17777546. When set to 0, the clock CSR is enabled. This parameter is normally set to 0.

G—Force Clock Interrupts

When set to 1, the clock unconditionally requests interrupts when the processor priority is 5 or less. When set to 0, the clock can request interrupts only if the clock CSR is enabled, clock CSR bit 6 is 1, and the processor priority is 5 or less. This parameter is normally set to 0.

NOTE

If the command parameter Force Clock Interrupts is selected, the user should always disable the clock CSR since the CSR has no control over the clock.

H—Clock Select

This parameter determines the source of the clock to be used. The choices are listed below.

Value	Source
0	Clock sourced from backplane pin BR1. The power supply normally drives this signal at 50 or 60 Hz.
1	Clock sourced on the KDJ11-BF at 50 Hz
2	Clock sourced on the KDJ11-BF at 60 Hz
3	Clock sourced on the KDJ11-BF at 800 Hz

I—Enable ECC Test

When set to 1, this parameter enables the ECC memory test to be run on any ECC memories present except UNIBUS memory. If the system contains a mix of ECC and non-ECC memory, the ROM code runs the ECC tests only on the ECC memories. The ROM code uses bit 4 of the memory CSR to determine if the memory is ECC or parity. If bit 4 is a read/write bit and can be written as a 1 and a 0, the ROM code assumes the memory is ECC. When this parameter is set to 0, the ECC test is always bypassed. This parameter is normally set to 1 even when ECC memory is not present. This parameter is reset if an ECC memory is installed whose ECC hamming code does not match that of an MS11-P type memory.

J—Disable Long Memory Test

When set to 1, this parameter bypasses the memory address shorts data test for all memory above 256 Kbytes. When set to 0, the address shorts data test is run on all available memory. This parameter is normally set to 0.

NOTE

If the long memory test is disabled and parity memory exists above 256 Kbytes, it is very likely that memory will contain parity errors after power-up.

K—Disable ROM

This parameter allows the user to selectively disable all or part of the ROM code after the selected device has been booted. Normally the ROM code on the CPU responds to two 256-word pages in the I/O page. One page responds to addresses from 17773000 to 17773777, the other page responds to addresses from 17765000 to 17765777. Both of these pages are automatically enabled at power-up or restart. After a device is booted, one or both of these pages may be disabled by the ROM code. The following table lists the variations of this parameter. This parameter is normally set to 0.

Value	ROM Pages Disabled
0	None
1	17765000–17765777
2	17773000–17773777
3	17765000–17765777 and 17773000–17773777

NOTE

If the ROM code is booting directly from a M9312-type boot ROM located on either the UBA module or the M9312 module, the ROM code automatically disables the CPU ROM in the 17773xxx address range and enables the ROMs on the board which were selected for booting. This action is taken regardless of the status of the disable UBA ROM parameter (Q) and the disable ROM parameter (K).

L—Enable Trap on Halt

If this parameter is set to 1, the processor traps to location 4 if a halt instruction is executed in kernel mode. If this parameter is set to 0, the processor enters J-11 micro ODT if a halt instruction is executed in kernel mode. This parameter is normally set to 0.

M—Allow Alternate Boot Block

After the boot block of a device is loaded into memory, the ROM code looks at word locations 0 and 2 to see if the device looks bootable. If the data is not correct, the ROM code types out an error message indicating that the Media is not bootable. When this parameter is set to 1, the ROM code looks for location 0 to be any non-zero number. If this parameter is set to 0, the ROM code looks for location 0 to be a value of 240 to 277 and for location 2 to be 400 to 777. This parameter is normally 0 but may have to be changed to 1 to allow some users' operating systems to boot properly.

N—Disable Setup Mode

If this parameter is set to 1, the user is not able to enter setup mode from dialog mode. The command lines in dialog mode will not show the setup command. If the user types the command, the response will be "invalid command."

If forced dialog mode is selected, then setup mode is unconditionally enabled regardless of the value of this parameter. This parameter allows some users to prevent unauthorized entry into setup mode. This assumes that the user has the forced dialog switch under some type of physical control.

O—Disable All Testing

If this parameter is set and forced dialog mode is not enabled, the ROM program bypasses virtually all testing. No location in memory is changed unless the selected boot program makes a change. This is a special parameter that should not be used unless necessary. It has been provided for cases where the user needs almost immediate response at power-up, and when the user needs the contents of memory to be left unaltered.

NOTE

If all testing is disabled and parity memory exists, then it is very likely that memory will contain parity errors after power-up.

P—Enable UNIBUS Memory Test

If this parameter is a 1, then any available UNIBUS memory is tested; if 0, UNIBUS memory is not tested. This parameter is normally a 1. If UNIBUS memory is not present, then no action is taken by the ROM code.

Q—Disable UBA ROM

This parameter is copied to bit 3 in the DCSR of the UBA after a normal boot. When this bit is 1, the UBA ROMs are disabled. This allows other ROM boards on the UNIBUS to show up in the UBA ROM address range of 17773000 to 17773776. When this bit is 1, the UBA ROMs are enabled. This bit is normally 0. When a user tries to boot either a UBA or M9312 ROM boot, this parameter is ignored.

NOTE

If the ROM code is booting directly from a M9312-type boot ROM located on the M9312 module, the ROM code automatically disables the CPU ROM in the 17773xxx address range and the ROMs on the UBA module. The ROMs on the M9312 module will be enabled. This action is taken regardless of the status of the Disable UBA ROM parameter (Q) and the Disable ROM parameter (K).

R—Enable UBA Cache

When a 1, this parameter causes the UBA cache to be enabled and to be tested by the ROM code. If a failure occurs during testing of the UBA cache, then it will be disabled. When 0, the UBA cache is always disabled and is not tested. This parameter is normally a 1.

S—Enable 18-Bit Mode

This bit is copied to bit 5 of the KMCR on the UBA. When a 1, this causes 18-bit addressing only. When a 0, 22-bit addressing occurs. This parameter is normally a 0.

3.3.3 Setup Command 3

This command prints out the current contents of the translation table and allows the translation table to be changed. The translation table is used to allow devices to be booted using nonstandard CSR addresses. When the ROM program enters the boot routine, R0 contains the unit number and R2 contains the device name (mnemonic). The ROM code tries to find a match in the translation table for the device name and unit number. If no match is found, the boot program uses the default CSR address for the device. If a match is found, the translation table defines the CSR address to be used.

Example 3-17 shows this command.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 3 [Return]

List/change boot translations in the Setup table

TT1     blank
TT2     blank
TT3     blank
TT4     blank
TT5     blank
TT6     blank
TT7     blank
TT8     blank
TT9     blank

Type Ctrl Z to exit or press the RETURN key for No change

TT1     blank
        Device name     =
```

Example 3-17 Setup Command 3

The ROM code is now waiting for the user to enter a new device name. If the user does not want to change any items in the translation table he/she would type [Ctrl] [Z]. This returns the user to the setup mode prompt. The user may skip over any entry and go to the next entry by pressing [Return]. To enter a new device or change an entry, the user types the new device the unit number, and the CSR address.

In Example 3-18, the user has a system that has one RA80 and an RA60 using a UDA50 controller at the standard address of 172150. The user also has an RC25 with a KLESI-U controller. Since the UDA50 and the KLESI-U share the same standard CSR address, one of them must be set to respond to a different address. In this example, the KLESI interface is set to respond to address 17760500. The RC25 has a unit number plug set for units 4 and 5. The RA80 is unit 0 and the RA60 is units 1 and 2. Since the RC25's interface is at a nonstandard CSR address and there are two unit numbers, there will be two entries in the translation table for units 4 and 5.

```
TT1      blank
Device name      = DU [Return]
Unit number      = 4 [Return]
CSR address      = 17760500 [Return]
TT1    DU4 address 17760500

TT2      blank
Device name      = DU [Return]
Unit number      = 5 [Return]
CSR address      = 17760500 [Return]
TT2    DU5 address 17760500

TT3      blank
Device name      = [Ctrl] [Z]
```

Example 3–18 Two-Entry Translation Table

The translation table also provides a means of handling multiple controllers such as RL02 controllers. For example, if the user has two RL02 controllers with six drives of which two are on the second controller at address 17760400, the translation table could be set up to handle this. Drives 0–3 would be on the first controller at the standard address and would not require any entries. Drives 0 and 1 on the second controller would be labeled as drives 4 and 5 and entered into the translation table. Since RL02 controllers only recognize unit numbers from 0–3 the unit numbers 4 and 5 would have to be translated to unit numbers 0 and 1. Example 3–19 shows a translation table with entries.

```
TT1      blank
Device name      = DL [Return]
Unit number      = 4 0 [Return]
CSR address      = 17760400 [Return]
TT1      DL4 = DL0 address 17760400

TT2      blank
Device name      = DL [Return]
Unit number      = 5 1 [Return]
CSR address      = 17760400 [Return]
TT2      DL5 = DL1 address 17760400

TT3      blank
Device name      = [Ctrl] [Z]
```

Example 3–19 Unit Number Translation

3.3.4 Setup Command 4

This command allows the user to select the devices to be tried in the automatic boot sequence. The user creates a small list that defines the devices and the order in which they are to be tried. One entry is needed to define a device and its unit number. If the same device is used more than once with different unit numbers, then one entry is needed for each unit number.

NOTE

The selections for this command use the same locations in the EEPROM as command 6 that follows.

When command 4 is executed, the ROM code prompts the user for a device name. The user then types either the single- or double-letter mnemonic associated with the device to be selected. The ROM code then prompts for the unit number. The ROM code continues prompting for all six entries in the table.

Example 3-20 shows command 4 being executed. In this example, the user adds the boot for the RX02 unit 1 (DY) by replacing the exit name (E) with DY and typing in the unit number next. The exit name is typed for the next entry.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 4 [Return]

List/change the Automatic boot selections in the Setup table

A = Disk MSCP automatic boot
B = External ROM boot
E = Exit automatic boot
L = Loop continuously

Boot 1 = A
Boot 2 = DL0
Boot 3 = MS0
Boot 4 = MU0
Boot 5 = E
Boot 6 = blank

Type Ctrl Z to exit or press the RETURN key for No change

Boot 1 = A
Device name     = [Return]

Boot 2 = DL0
Device name     = [Return]

Boot 3 = MS0
Device name     = [Return]

Boot 4 = MU0
Device name     = [Return]

Boot 5 = E
Device name     = DY [Return]
Unit number     = 1 [Return]

Boot 6 = blank
Device name     = E [Return]
Unit number     = 0 [Return]

KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key:
```

Example 3-20 Setup Command 4

Table 3-2 lists the four special single-letter mnemonic device names and the associated ROM action.

Table 3-2 ROM Code Action

Mnemonic	Description
A	The ROM code boots the first bootable disk MSCP device it can find. The ROM code tries removable media units first, then fixed media units.
B	This mnemonic causes the ROM code to check for a UNIBUS ROM board in address range of 17773000 to 17773776. If the ROM exists and location 17773024 is not odd, the ROM code disables the internal CPU ROMs and the UBA ROMs and jumps to the location specified in location 17773024 of the UNIBUS ROM board. Usually this board is an M9312, M9301, or user-supplied equivalent.
E	The only purpose of this mnemonic is to indicate to the ROM code that there are no other devices to try in the list. This is used when there are five or fewer devices in the list. It follows the last device in the list to be tried. If all six entries are filled in the list, then this mnemonic is "not needed"; the list terminates automatically after trying the last entry. When this mnemonic is reached, the ROM code restarts the boot sequence from the beginning and prints error messages for each device that fails to boot as it is tried. After the second pass through the list without successfully booting a device, the ROM code enters dialog mode.
L	This mnemonic also marks the end of the list. When this mnemonic is reached, the ROM code restarts the boot sequence at the beginning of the list. It continues trying to boot each device in the list until either a successful boot has occurred, or the sequence is terminated by the user typing Ctrl C. All boot error messages are suppressed when the ROM code is looping on the boot list.

The action taken by the ROM code for the four single-letter mnemonics A, B, E, and L applies only to automatic boot mode with the exception of B which can also be executed from the dialog mode Boot command. The dialog mode Boot command treats the single-letter mnemonics A, E, and L as invalid devices if they are used as the device name in the Boot command. If the user creates a bootstrap program and loads it into the EEPROM, that program should not be given a device name of A, B, E, or L since these are already defined. All other single-letter mnemonics are open for use.

3.3.5 Setup Command 5

This command allows the user to store up to 20 bytes of information in the EEPROM. The data must be entered in octal numbers in tha range of 0 to 377. The setup mode initialize command resets this data to 0.

3.3.6 Setup Command 6

This command allows the user to define the value of three of the eight switches at the edge of the CPU module in order to boot specific devices. This command defines six of the eight possible combinations of switches 2-4. The other two combinations have a fixed definition that cannot be changed. See Appendix C for additional information.

Command 6 is not normally used. The only time these switches might be defined is:

- If the system has cabling between J3 on the CPU module and a remote 8-position rotary switch
- The user wants to define six positions such that each position causes the ROM code to enter automatic boot mode after testing is completed and attempts to boot only one device which was defined by this command in setup mode.

Normally the three switches are off. If automatic boot mode is selected, the ROM code uses the list defined by command 4 in setup mode and tries to boot each item in the list until all items have been attempted.

When switches 2-4 are set to one of the six combinations shown in the example (Example 3-21) and forced dialog mode is not selected, the ROM code enters the auto boot mode and attempts to boot only the one device selected by this command. If the boot is unsuccessful, the ROM code displays the normal error message and enters dialog mode.

NOTE
The six selections in the table are stored in the same area of the EEPROM as the six boot selections described in command 4.

Example 3-21 shows setup mode command 6 with three of the six possible positions being defined to select DU0, DU1, and DU2. The other three are defined to select DL0.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 6 [Return]

List/change the switch boot selections in the Setup table

Switches 2,3,4  on   on   off   = DU0
Switches 2,3,4  on   off  on    = DU1
Switches 2,3,4  on   off  off   = DU2
Switches 2,3,4  off  on   on    = DL0
Switches 2,3,4  off  on   off   = DL0
Switches 2,3,4  off  off  on    = DL0

Type Ctrl Z to exit or press the RETURN key for No change

Switches 2,3,4  on   on   off   = DU0
Device name     =
```

Example 3-21 Setup Command 6

3.3.7 Setup Command 7

This command performs the same function as the List command in dialog mode. The command is duplicated in setup mode for user convenience. See Section 3.2.3 for a detailed description of this command. Setup mode is restarted at the completion of this command.

3.3.8 Setup Command 8

This command initializes the current contents of the setup table in memory to the default values. This command does not affect the contents of the EEPROM itself. The setup Save command (command 9) must be executed in order to save the setup table into the EEPROM. Command 9 only affects parameters associated with commands 2 to 6 of setup mode. It does not affect any data that is not in the first 105 bytes of the EEPROM.

The values of the parameters after command 8 is entered are as follows:

- All parameters listed under Setup command 2 of setup mode are set to 0 with the exception of A, B, C, I, P, and R, which are set to 1, and E, which is set to 7. (See Figure 3-16.)
- All entries in the translation table under Setup command 3 are cleared and will list as blank.
- The automatic boot selection list under Setup command 4 will be set to A, DL0, MS0, MU0, E, blank.

Since Setup command 6 shares the same area as command 4, its list of parameter values are identical to that of command 4.

To enter this command, the user types 8 Return. After the ROM code prompt, the user types 1 Return. Command 9 (SAVE) should be executed after command 8 if the user wishes to retain the defaults in the EEPROM. Example 3-22 shows command 8 being executed.

```
KDJ11-B Setup mode

Press the RETURN key for Help
Type a command then press the RETURN key: 8 Return

Initialize the Setup table

Are you sure ?  0=No,    1=Yes
Type a command then press the RETURN key: 1 Return

KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key:
```

Example 3-22 Setup Command 8

3.3.9 Setup Command 9

This command copies the current contents of the setup table in memory into the EEPROM. The command should be executed after any changes are made to the EEPROM in setup mode. This is the only command that writes anything into the first 105(10) bytes of the EEPROM. When saving data into the EEPROM, the ROM code writes only the locations that need to be written.

This command always writes a new and correct checksum into the EEPROM unless a failure occurs. If a location cannot be written, the ROM code tries once more and then reports the error. It takes approximately 15 ms to write each location.

If command 9 is entered and no changes have been made to the setup table, the ROM code displays a message saying no changes were made. Command 9 then restarts setup mode. If changes are to be made, the ROM code prompts the user to make sure the user wants to make the changes. Example 3-23 shows command 9.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 9 Return

Save the Setup table into the EEPROM

Are you sure ?  0=No,   1=Yes
Type a command then press the RETURN key: 1 Return

Writing the EEPROM

KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key:
```

Example 3-23 Setup Command 9

3.3.10 Setup Command 10

This command restores the setup table in memory with the values actually stored in the EEPROM. This command allows the user to restore the setup table after making some temporary changes. It is also used to load the actual data from the EEPROM into the setup table if an error occurred during the EEPROM checksum tests.

When an error occurs during the EEPROM checksum tests, the ROM code assumes the data is bad and loads a set of default values into the setup table and uses them. In this case, the user could load the actual data and then verify the data before trying to save it back into the EEPROM.

Example 3-24 shows command 10.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 10 Return

Load EEPROM data into the Setup table

Are you sure ?  0=No,   1=Yes
Type a command then press the RETURN key: 1 Return

KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key:
```

Example 3-24 Setup Command 10

3.3.11 Setup Command 11

This command allows the user to delete an EEPROM boot. If this command is executed, the ROM code prompts the user for the device name of the EEPROM boot to be deleted. After the device name is typed, the ROM code looks for the first boot program in the EEPROM with that device name. The ROM code deletes the boot if found. If there are any boot programs following the deleted program, the ROM code automatically moves all of these programs up to use the space made available by the deleted program (see Example 3-25).

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 11 [Return]

Delete an EEPROM boot

Type Ctrl Z to exit or press the RETURN key for No change

Device name     = CC (RETURN)

Are you sure ?  0=No,    1=Yes
Type a command then press the RETURN key: 1 [Return]

KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key:
```

Example 3-25 Setup Command 11

3.3.12 Setup Command 12

This command is used to copy an EEPROM boot program into memory. When the command is executed, the ROM code prompts the user for the device name of the EEPROM boot program to be loaded in memory. The program can then be examined and/or edited using SETUP command 13. Example 3-26 shows command 12.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 12 [Return]

Load an EEPROM boot into memory

Type Ctrl Z to exit or press the RETURN key for No change

Device name     = CC (RETURN)

Are you sure ?  0=No,    1=Yes
Type a command then press the RETURN key: 1 [Return]

KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key:
```

Example 3-26 Setup Command 12

3.3.13 Setup Command 13

This command is used to either create a new EEPROM boot program or to edit a program previously loaded with command 12. Command 13 allows the user to change the device name, the device description, the allowable unit number range, the beginning and ending addresses of the program in memory, and the start address of the program.

When these changes are complete, the ROM code enters ROM ODT which is a ROM code version of J-11 micro ODT. When this command is first entered, it will list the available space in the EEPROM for bootstrap programs.

Example 3-27 shows command 13.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 13 Return

Edit/create an EEPROM boot

Type Ctrl Z to exit or press the RETURN key for No change

1410 Bytes free in the EEPROM

Device name            = AA          New = EA Return

Beginning  address     = 000600      New = 10000  Return

Last byte address      = 000615      New = 10177  Return

Start address          = 000600      New = 10000  Return

Highest Unit number    = 3           New = 255  Return

Device Description     = EA BOOT     New = RM02,RM03  Return

Enter ROM ODT

xxxxxx/ = open word location xxxxxx if address even, byte if odd
RETURN  = close location
. or LF = close location and open next
-       = close location and open previous

ROM ODT> 010000/000000 012705 Return
ROM ODT> 010002/000000 101 Return
ROM ODT> 010004/000000 12706 Return
ROM ODT> 010006/000000 1000 Return
etc.     Type Ctrl Z to exit back to the Setup mode menu.
```

Example 3-27 Setup Command 13

The beginning address is the first location of the program in memory. The last byte address is the address of the last byte of code used in memory. If in doubt, use the last address of data plus 2 for this value. Do not use a much larger number since it will waste EEPROM space.

The start address is the address to which the ROM code will pass control. The start address does not have to be the same as the load address, but it must be even and a value in the range defined by the load and ending addresses.

The highest unit number defines the allowable range of valid unit numbers for this device. If the value is set to 3, the allowable range is 0 to 3. The highest range is 0 to 255. If a unit number is typed in at boot time and it is not in range, then an invalid unit number error occurs.

The device description is an optional but recommended description of the device name. The maximum length of this name is 11 characters or spaces. The name should normally be the name that is physically marked on the outside of the device (for example, RL02).

3.3.14 Setup Command 14

This command allows the user to save the existing boot program located in memory in the EEPROM. This is the only command that actually writes a boot into the EEPROM. The other commands only change a copy of the boot program that resides in memory.

When saving a boot program into memory, the device name of the program must not match the name of an existing program in the EEPROM. If the program name already exists, the user must delete that program first or change the name of the program to be saved. If two or more programs were written into the EEPROM with the same name, only the first one would be found and used. Example 3-28 shows command 14.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 14 [Return]

Save boot into the EEPROM

Type Ctrl Z to exit or press the RETURN key for No change

Are you sure ?  0=No,   1=Yes
Type a command then press the RETURN key: 1 [Return]

Writing the EEPROM - Please wait

KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key:
```

Example 3-28 Setup Command 14

Example 3-29 shows command 14 being executed. In the example, the data in the setup table matches the data in the EEPROM, thus no changes are made.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 14 [Return]

Save boot into the EEPROM

Boot is already in the EEPROM

No changes made

KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key:
```

Example 3-29 Setup Command 14 with No Changes

3.3.15 Setup Command 15

This command puts the user into ROM ODT (Example 3-30). The ROM code opens up the address defined by the beginning address of the program. ROM ODT is not the same as J-11 micro ODT. The only purpose of ROM ODT is to allow the user to create or edit a small bootstrap program to be stored in the EEPROM.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 15 [Return]

Enter ROM ODT

Type Ctrl Z to exit or press the RETURN key for No change

xxxxxx/ = open word location xxxxxx if address even, byte if odd
RETURN  = close location
. or LF = close location and open next
-       = close location and open previous

ROM ODT> 010000/000000 012705 [Return]
ROM ODT> 010002/000000 101 [Return]
ROM ODT> 010004/000000 12706 [Return]
ROM ODT> Ctrl Z

KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key:
```

Example 3-30 Setup Command 15

In ROM ODT, the only allowable addresses that can be examined are the addresses of memory from 0-28 Kword (0-00157776). Any other addresses and any attempt to access the I/O page or any registers are not allowed. Table 3-3 lists the ROM ODT commands.

Table 3-3 ROM ODT Commands

Command	Symbol	Use
Slash	/	Prints contents of specified location. If no address is defined, prints contents of the last location that was opened. If location opened is an odd number, then prints out only the contents of the byte. If location is even, then mode is word; if location is odd, then mode is byte. Leading zeroes are assumed. Only bits <15:0> of the address are used.
Return	[Return]	Closes an open location.
Line Feed	[Line Feed]	Closes an open location and then opens the next location. If word, increment address by 2; if byte, increment address by 1.
Period	.	Alternate character for line feed. This command is useful when the terminal is a VT200 series terminal. It is also convenient to use with the keypad.
Up arrow	↑	Closes an open location and then opens the previous location. If in word mode then decrement by 2; if byte, decrement by 1.
Minus	–	Alternate character for up arrow. This command is useful when the terminal is a VT200 series terminal. It is also convenient to use with the keypad.
Delete	[Delete]	Deletes the previous character typed.
Ctrl/Z	^Z	Exit ROM ODT and return to setup mode.

The following sections present examples of ROM ODT use.

Example 1 Location 200 is opened. It is then closed with no changes and location 202 is opened. Location 202 is then closed after changing its contents.

```
ROM ODT > 200/
ROM ODT > 000200/100000 [Line Feed]
ROM ODT > 000202/003333 44 [Return]
ROM ODT >
```

Example 2 Byte location 1001 is opened. It is then closed and locations 1002 and 1003 are opened. Location 1003's data is changed and then closed.

```
ROM ODT > 1001/
ROM ODT > 001001/101 [Line Feed]
ROM ODT > 001002/104 [Line Feed]
ROM ODT > 001003/113 141 [Return]
ROM ODT >
```

Example 3 The user attempts to open location 170000 which is in the I/O page, and not allowed.

```
ROM ODT > 170000/
ROM ODT >
```

Example 4 Location 150000 is opened and then closed. It is then reopened by typing a slash (/) only.

```
ROM ODT > 150000/
ROM ODT > 150000/032737 [Return]
ROM ODT > /
ROM ODT > 150000/032737
```

3.4 Diagnostic Error Messages

When an error occurs, the test number and other information are printed. The error number is always that of the current test the ROM code is running. The test numbers and corresponding error messages are described in Chapter 5.

3.5 Bootstrap Programs

Bootstrap programs are found in various areas of the system. The ROM code contains bootstrap programs for the following UNIBUS devices:

Device Name	Device Type
DU[1]	RX50, RC25, RA80, RA81, RA60
DL	RL01, RL02
DX	RX01
DY	RX02
DD	TU58
DK	RK05
MU[2]	TU81

[1]DU refers to a general-purpose bootstrap program for disk MSCP devices.

[2]MU refers to a general-purpose bootstrap program for tape MSCP devices.

For users who have devices not covered by the ROM code bootstrap list, the ROM code also supports the use of M9312-type boot ROMs. These are typically shipped with the controller module for devices that can be booted. These ROMs are currently used in all existing UNIBUS PDP-11 products. For example, an RK07 disk drive uses an M9312 ROM to allow the drive to be booted. See Table 3-4 for a list of the M9312-type boot ROMs currently available.

The user may install the ROM in either the UBA module or an optional M9312 module. The ROM code lists and boots any M9312-type ROM located in either module. Section 3.5.1 contains additional information on M9312-type ROMs.

The ROM code also allows users to install bootstrap programs for new devices or for custom boot programs in the EEPROM. The user can install machine language programs into the EEPROM by using setup mode commands 11 through 15. Once the program is loaded into the EEPROM, it will be available to the user at any time.

There is one important difference between boot programs in the EEPROM and others. The EEPROM is an 8-bit device. Programs, therefore, cannot be executed out of the EEPROM as they are from the ROM code or M9312-type ROMs. The ROM code always assembles EEPROM boot programs in memory. The code then starts the program at a start address defined by the boot program. The EEPROM may contain more than one boot program, depending on the size of the programs.

When a boot program is requested, the ROM code searches for the boot program in the following sequence.

1. Search the EEPROM on the CPU first.
2. Search the ROM code on the CPU next.
3. Search the ROM sockets on the UBA.
4. Search the ROM sockets on the M9312 board, if present.

3.5.1 Bootstrap List

Table 3-4 describes the M9312-type boot ROMs that are available.

These ROMs are used on all UNIBUS processors. Some of the ROMs listed are not required because the base ROM code in the CPU ROM contains bootstraps for some devices.

Many of the devices listed are old and generally not available. However, they are included because many systems contain these devices. In general, the ROMs needed to boot a device are shipped with the interface for the device itself. For example, if a customer purchased an RL02 and an RL11 controller the ROM would come with the RL11 controller.

The CPU ROM code uses a two-letter mnemonic ROM identifier contained in each M9312 ROM. Some of the ROMs contain more than one bootstrap program. Some programs require more than one ROM to fully implement a bootstrap.

Table 3-4 Available M9312-Type ROM

Mnemonic	Part Number	Supported Devices
CT	23-761A9-00	TU60 cassette tape drive.
DT	23-756A9-00	TU55, TU56 tape drives.
DM	23-752A9-00	RK06, RK07 disk drives.
DP	23-755A9-00	RP02, RP03 disk drives.
DB	23-755A9-00	RP04, RP05, RP06, RM02, RM03 disk drives.
DS	23-759A9-00	RS03, RS04 disk drives.
MM	23-757A9-00	TU16, TE16, TU45, TM02, TM03, TU77 tape drives.
MS	23-764A9-00	TS04, TS11, TU80, TS05 tape drives.
MT	23-758A9-00	TU10, TE10, TS03 tape drives.
PR	23-760A9-00	PC05 high-speed paper reader.
TT	23-760A9-00	Low-speed paper reader (Teletype).
XE	23-E22A9-00	DECnet DEUNA Ethernet interface.
XL	23-926A9-00 23-927A9-00 23-928A9-00	DL11-E (DECnet DDCMP).*
XM	23-862A9-00 23-863A9-00 23-864A9-00	DMC11, DMR11 (DECnet DDCMP).*
XU	23-868A9-00 23-869A9-00 23-870A9-00	DU11 (DECnet DDCMP).*

*Three ROMs are required to implement this bootstrap.

Table 3-4 (Cont.) Available M9312-Type ROM

Mnemonic	Part Number	Supported Devices
XW	23-865A9-00 23-866A9-00 23-867A9-00	DUP11 (DECnet DDCMP).*

*Three ROMs are required to implement this bootstrap.

NOTE

In V6.0 and V7.0 ROM code, the routine that identifies ROMs on the UBA or the M9312 does not correctly identify boot ROMs that use more than one ROM for the boot. Because of this, these boots do not list and cannot be started from the base ROM code without using a small EEPROM boot program to transfer control to the ROMs. This problem is corrected in version 8.0 of the CPU ROM code. (See Appendix E.)

See Appendix F for instructions on how to set up the EEPROM to allow the CPU ROM to handle a multi-ROM boot on the UBA or the M9312, or any ROM which is not compatible with the M9312 ROM format for boot ROMs.

The ROMs listed in Table 3-5 are generally not needed because similar bootstrap programs are located in the base ROM on the CPU module.

Table 3-5 CPU ROM Boot Programs

Mnemonic	Part Number	Supported Devices
DD	23-765A9-00	TU58 cartridge tape drive
DK	23-756A9-00	RK05 disk drive
DL	23-751A9-00	RL01, RL02 disk drives
DU	23-767A9-00	(General boot for all disk MSCP devices) RA80, RA81, RA60, RC25, RX50 disk drives
MU	23-E39A9	(General boot for all tape MSCP devices) TR50, TU81
DX	23-753A9-00	RX01 floppy disk drive
DY	23-811A9-00	RX02 floppy disk drive

3.5.2 EEPROM Format

The first 105 bytes of the EEPROM store the base hardware parameters for the CPU and the UBA. Also in these first 105 bytes is the information needed by the ROM code to determine the power-up/restart mode, test selections, and the list of devices to try to boot. Immediately following the first 105 bytes are four that the user may use for any purpose, such as storing a serial number. These four bytes are never used by the ROM code.

The optional bootstrap programs follow immediately after these four bytes. The EEPROM may also contain translations for some of the ROM code messages for local language requirements for non English users. The local language text, if present, always starts at the end of the EEPROM. Figure 3-1 illustrates the EEPROM layout.

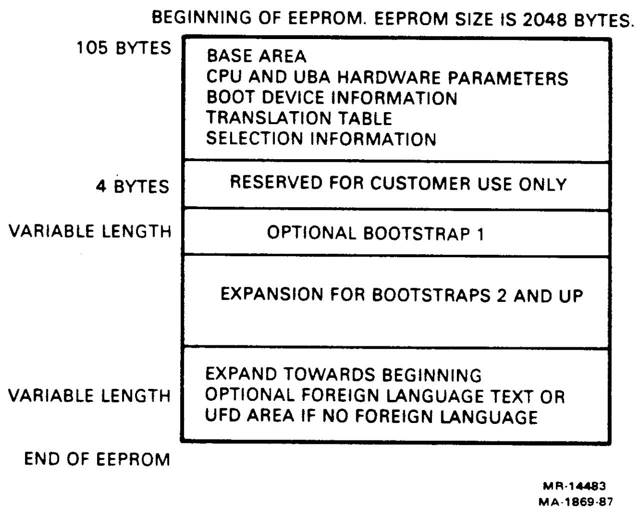
BEGINNING OF EEPROM. EEPROM SIZE IS 2048 BYTES.
105 BYTES
BASE AREA
CPU AND UBA HARDWARE PARAMETERS
BOOT DEVICE INFORMATION
TRANSLATION TABLE
SELECTION INFORMATION
4 BYTES
RESERVED FOR CUSTOMER USE ONLY
VARIABLE LENGTH
OPTIONAL BOOTSTRAP 1
EXPANSION FOR BOOTSTRAPS 2 AND UP
VARIABLE LENGTH
EXPAND TOWARDS BEGINNING
OPTIONAL FOREIGN LANGUAGE TEXT OR
UFD AREA IF NO FOREIGN LANGUAGE
END OF EEPROM
MR-14483
MA-1869-87

Figure 3-1 EEPROM Layout

The four bytes reserved for the user may be accessed as follows:

1. Make sure bit 6 is reset in the CPU BCSR at 17777520.
2. Make sure bit 5 is set in the CPU BCSR at 17777520.
3. Bit 4 in the CPU BCSR at 17777520 must be set to write the EEPROM. It does not need to be set when reading the EEPROM.
4. The low byte of the PCR must be 0 at 17777522.
5. The four bytes of data can now be read or written at addresses 17765322 to 17765330. Each byte is accessed in a 16-bit word where the data is in bits 7 through 0. Bits 15 through 8 are not driven by the CPU and must be ignored since their value will change.

If a serial number is written to the four user-reserved bytes, the user could, for example, write a number up to 24-bits long in the first three bytes and write a checksum for the number in byte 4. The actual format is up to the user.

When data is written to the EEPROM, the user must wait at least 10 ms after the write cycle for each byte before proceeding. The EEPROM cannot be written more than 10,000 times. After any write, the data should always be checked after 10 ms to verify that it was written correctly.

NOTE

It is strongly recommended that bit 4 of the BCSR not be set unless the user is writing to the EEPROM. Bit 4 should be reset as soon as writes are complete. This bit does not have to be set to read the EEPROM.

3.5.3 General Rules for EEPROM User Boots

EEPROM boots are assembled into the lower 28 Kwords of memory by the ROM code. If there are no errors, such as checksum errors, the ROM code passes control to the program according to the starting address in the boot program.

At the start of an EEPROM boot, the following is true.

- Memory management is disabled and 22-bit mode is off.
- R0 contains the unit number.
- R1 is 0 if no address was passed by the translation table or the /A switch in the Boot command. R1 is an alternate address for the program to use if the translation table matches the boot device's name and unit number with an entry in the translation table or the /A switch was used in the Boot command.
- The ROM code loads a trap handler for timeouts to location 4 in memory. The program is loaded into memory starting at location 1000 if the start address of the EEPROM boot program is 10000 or greater. The program is loaded into memory starting at location 17600 if the start address of the EEPROM boot program is 0 to 7776. The ROM code loads the address of the timeout handler into location 4 as long as the EEPROM boot does not already occupy location 4 itself. If a timeout occurs during the boot program and the timeout handler is entered, the handler restarts the ROM code with the nonexistent controller message in R5; the ROM code will assume that the value in R1 is the address of the controller that timed out.

The EEPROM may use memory management and restart the ROM code if the EEPROM program restricts the use of MMU to only kernel instruction PAR/PDRs 0-3 and 7. The EEPROM boot must not use any of the other PAR/PDRs if the ROM code is to be restarted. The ROM code should be restarted with MMU OFF (MMR0 bit 0 = 0). In the case where the EEPROM boot overlaps location 4, it is the responsibility of the EEPROM boot to handle timeouts.

The user's program is responsible for booting the device and checking the boot block for a bootable secondary boot program.

It is recommended that EEPROM boot strap programs not be located in memory between addresses 2000 to 2300, 16000 to 16040, and 20000 to 40000. The recommended location for EEPROM bootstrap programs is memory addresses above the 8 Kword point (physical address 00040000).

The following sections describe how user-written boot programs handle errors or success. It is recommended that the user writes programs which adhere to these rules. The user will then receive meaningful messages in the event errors occur in the boot program. However, the user's boot program need not return control if it is not desired.

Once the EEPROM boot is started, the user's boot program has complete control of the CPU. The user's program would restart the ROM code for one of the following three reasons:

- An error occurred when a boot of a device was attempted. The ROM code is restarted to type out a general error message. This allows the automatic boot mode to try another item for booting.

  To re-enter the ROM code for error message printing, the user's boot program loads R5 with the error message desired, ensures that bits <7:4> of the BCSR are set to 0, and then executes a JMP @ 165762 with the MMU off.

  The following list specifies the octal value of the error message selection codes and the text of the message printed. At the time the ROM code is restarted, R0 should still contain the unit number and R1 should contain the address of the controller.

  270—Drive not ready
  272—No disk present or drive unloaded
  273—No tape present
  274—Non existent controller
  275—Non existent drive
  276—Invalid unit number
  277—Invalid device
  300—Controller error
  301—Drive error

- If the user's program monitors the keyboard during the boot, it could return control to the ROM code if Ctrl C is pressed. This is an optional feature. The ROM code would be re-entered the same way as described in item 1 above, except that R5 should be an octal value not equal to 270 to 301 or 1.

- The boot is successful and the ROM code is temporarily restarted to print out the "Starting system from" message. After the message printout is complete, the ROM code returns control to the user's program. To re-enter the ROM code in order to print the "Starting system" message, the user's code would load R5 with the number 1, make sure bits <7:4> of the BCSR are 0; and then restart the ROM code by executing a JSR PC, @ 165762 with MMU off.

  When the ROM code has completed typing the message, it returns control to the user's boot at the instruction following the JSR instruction. At this point, to be compatible with all existing versions of the ROM code, the user's program should do the following.

  - Reset bit 11 (register set 1 select) in the PSW.
  - Reset the display register by writing 000077 to address 17777524.
  - Clear MMR3 (17772516) to make sure 22-bit mode is off.

3.6 Boot ROM Facility (M9312 compatible)

The KTJ11-B boot ROM facility allows the user to install M9312-compatible boot programs written for UNIBUS devices not directly supported by the KDJ11-BF boot programs. ROM programs that run on the M9312 should run on the KTJ11-B. The M9312-compatible boot programs are implemented in one to four 512 by 4-bit ROMs. Each ROM contains 64, 16-bit words of accessible code which are located in the first half of the ROM. The last 256 4-bit ROM locations are not used.

The KDJ11-BF CPU module can be configured to boot the system from one of its self-contained boot programs, from the KTJ11-B boot ROM facility, or a boot ROM option which resides on the UNIBUS.

3.6.1 Boot ROM Installation

M9312-compatible boot ROMs must meet Digital Equipment Corporation purchase specification 23-000A9-01 for 512 × 4 tri-state PROMs. The ROMs must be encoded per Digital Equipment Corporation specification K-SP-M9312-0-8.

A single ROM may contain one or more boot programs. Alternatively, a single boot program may require up to four ROMs. However, ROM programmers have been encouraged to keep each boot program within a single ROM.

Table 3-6 presents the IC location of each ROM socket and its address range.

Table 3-6 ROM Locations and Addresses

ROM Number	IC Location	Addresses
1	E145	17773000—17773176
2	E144	17773200—17773376
3	E143	17773400—17773576
4	E142	17773600—17773776

3.6.2 ROM Addresses 17773000—17773776

The KTJ11-B boot ROM logic responds to addresses 17773000 through 17773776. As shown in Table 3-6, these locations are subdivided into four 64-word segments, each of which addresses one of the four ROM sockets. The ROM logic responds to a read operation by decoding a 16-bit data word from four successive locations in the selected 512 × 4 bit ROM. If an empty ROM socket is accessed, the data word read is typically 161777.

3.6.3 ROM Formats

The format information is contained in the M9312 specification K-SP-M9312-0-8. In summary, two types of ROM formats are identified: the format for one or more programs contained in a single ROM, and the format for one boot program contained in two or more ROMs. Specific information is given on the program headers and on the format of data within the ROMs.

3.6.4 Single ROM Programs

ROMs containing one or more programs in a single ROM conform to the following format.

- Each boot program must begin with a program header block as described in Section 3.5.2. The header block for the first (or only) program must start at ROM word address 0.
- Word address 24 of the ROM must remain reserved and be set to 173000.
- Word address 26 of the ROM must remain reserved and be set to 000340. (This location is required for systems that reboot via the M9312 module; it is not used by KTJ11-B systems that reboot via the CPU module.)
- Word address 176 of the ROM must be a CRC-16 word for the previous 63 words (omitting location 24 octal).

3.6.5 Multiple ROM Programs

Some boot programs require more than a single ROM. The format of the first ROM is described in Chapter 3. The continuation ROM(s) would have the following format.

- The first word of each continuation ROM must contain 177776 octal.
- Word address 24 of the ROM must remain reserved and be set to 173000.
- Word address 26 of the ROM must remain reserved and be set to 340. (This location is required for systems that reboot via the M9312 module; it is not used by KTJ11-B systems that reboot via the CPU module.)
- Word address 176 of the ROM must be a CRC-16 word for the previous 63 words (but omitting the word in location 24).

3.6.6 Program Header

The beginning of each boot program must contain a header section (see Section 3.6.4). This header section is formatted as follows.

- The first word contains an ASCII identifier. The identifier consists of two characters with a zero parity bit. The high and low bytes contain the first and second characters respectively. The characters are used by the KDJ11-BF boot and diagnostic ROM programs to search for the selected boot program. The KTJ11-B ROM code requires that both of these bytes contain characters with ASCII octal values of 101 to 132, or 141 to 172 (A-2 or a-2).
- The second word contains an offset from its address to the start of the next program header. If the ROM contains only one header, the second word, located in ROM address 2, contains 176, which points to the start of the next ROM.
- The third word address is the power-up entry point for unit 0. The third word address is used by systems containing the M9312 to disable a branch to diagnostics prior to running the boot program. KTJ11-B systems do not use this entry point.
- The fourth word address is an alternative entry point for unit 0, used by systems containing the M9312 to enable a branch to diagnostics before running the boot program. KTJ11-B systems do not use this entry point.
- The fifth word contains 0, indicating unit 0 for the instruction located in the previous word.

- The sixth word address is the entry point used by systems containing the M9312 for unit numbers other than 0. KDJ11-BF uses this entry point after first loading the unit number (zero or non-zero) into R0 and then setting the PSW carry bit (disabling the branch to M9312 diagnostics).
- The seventh word contains the address of the control status register of the device to be booted. It is used by the instruction located in the previous word.
- The eighth word contains an instruction that saves the PC in R4 for systems that branch to diagnostics on the next instruction. The KDJ11-BF uses this entry point when a nonstandard CSR address is passed in R1.
- The ninth word contains an instruction that branches to M9312 diagnostics if the PSW carry bit is clear. KTJ11-B systems always set the carry bit.
- The tenth word contains an unconditional branch to the start of the boot program.

3.6.7 ROM Data Organization

Each 512 × 4 bit ROM contains 64, 16-bit words of accessible code which are located in the first half of the ROM. Each 16-bit word is stored in four successive ROM locations. Whenever the KTJ11-B ROM logic is addressed by a data-in operation, it constructs the 16-bit word from the appropriate ROM locations. Table 3-7 presents the location and polarity of bits within each group of four nibbles. Note that bits 12, 11, and 10 are inverted, and bits 08 and 00 are not stored in the expected order.

Table 3-7 ROM Data Organization

	Data Bit 3	Data Bit 2	Data Bit 1	Data Bit 0
Nibble Number 0	03	02	01	08
Nibble Number 1	07	06	05	04
Nibble Number 2	11*	10*	09	00
Nibble Number 3	15	14	13	12*

*Data word bits 12, 11 and 10 are stored inverted.

3.7 J-11 Micro ODT

J-11 micro ODT is entered anytime the CPU is halted by:

- Placing the Restart/Run/Halt switch in the Halt position
- Executing a HALT instruction, if halt mode is enabled and the system is in kernel mode
- Pressing the Break key, if the terminal is set up to generate a break character and the front panel keylock switch is set to Enable.

J-11 micro ODT allows the user to examine or change any location in the 22-bit CPU memory space or I/O page memory space. In addition, J-11 micro ODT allows the user to start a program, to reinitiate program execution, and to single-step a program when the front panel switch is in the Halt position. Table 3-8 summarizes the J-11 micro ODT commands.

Table 3-8 J-11 Micro ODT Commands

Command	Symbol	Description
Slash	n/	Opens the specified location (n) and outputs its contents. n is an octal number.
Carriage Return	Return	Closes an open location.
Line Feed	Line Feed	Closes an open location and then opens the next contiguous location.
Internal Register	$n or Rn	Opens a specific processor register (n). n is an integer from 0 to 7 or the character S.
Processor Status Word Designator	S	Opens the processor status register. Must follow the $ or R command.
Go	G	Starts program execution.
Proceed	P	Resumes program execution.
Binary Dump	Ctrl Shift S	Manufacturing use only.

The following sections describe each J-11 micro ODT command specified in Table 3-8. In most cases, examples are given. Note that user input is in bold.

When entering addresses or data, leading zeros are not required. They are filled by ODT.

When entering addresses in the I/O page, all 22 bits must be entered (for example, 17776100).

A question mark (?) will be printed whenever illegal characters are entered, addresses are accessed that result in a timeout, or a parity error is detected.

3.7.1 / (ASCII 057) Slash

This command is used to open a memory location, I/O device register, internal processor register, or processor status word register. It must be preceded by octal digits to specify a location, or a register designator.

In response to the slash (/), ODT prints the contents of that location (that is, six characters) and waits for either data 33 for that location to be entered or a valid close command, (that is, Return or Line Feed). The slash (/) may be entered again immediately to display the contents of the previously opened location.

Examples

```
@1000/ 012737 [Return]         ;Open memory location 00001000.
                               ;The contents (012737) are
                               ;displayed. (RETURN) closes the
                               ;location without modification.

@100/ 000200 7422 [Return]     ;Open memory location 00000100
                               ;and deposit data (7422) and
                               ;close the location.

@/ 007422 6422 [Return]        ;Reopen the location and deposit
                               ;new data.
```

3.7.2 [Return] (ASCII 015)

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

Examples

See previous examples.

3.7.3 [Line Feed] (ASCII 012)

This command is the same as [Return] except the open location is closed and the next contiguous location is opened. Memory addresses are incremented by 2 and processor registers are incremented by 1. If the PS is opened, it is closed and no new location is opened.

Examples

```
@1000/ 012737 [Line Feed]        ;Location 1000 is opened, the
                                 ;contents are displayed, and
                                 ;then closed with (Line Feed).

00001002 100200 0 [Line Feed]    ;The (Line Feed) caused the next
                                 ;location to be opened and the
                                 ;contents to be displayed. In
                                 ;this case, the contents are
                                 ;changed by the operator.

00001004 176100 [Return]         ;The next location is opened
                                 ;to examine the contents and
                                 ;then closed with (Return).
```

3.7.4 $ (ASCII 044) or R (ASCII 122) Internal Register Designator

Either character, when followed by a register number 0 to 7 or the processor status (PS) designator S, opens that specific processor register. If more than one number is typed after the R or $, the last number typed is used.

NOTE

The trace bit (bit 4) of the PS cannot be modified by the user. This is because ODT uses the T-bit for single-stepping. The register set used with the R command is determined by PS 11. If PS 11 is set to a 1, register set 1 is used. If PS 11 is set to 0, register set 0 is used. The SP (R6) that is used is determined by PS<15:14>.

3.7.5 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. The program counter (PC) (R7) is loaded with the address (0 is used if no data is entered), and the following registers are cleared to zero: PS, MMR0<15:13,0>, MMR3, PIRQ, CPU error register, memory system error register, cache control register, and floating-point status register. Cache is flushed, and the UNIBUS is initialized.

If the G command is issued with the front panel switch in the Halt position, the system is initialized, ODT is reentered, and the PC is displayed. The GO command truncates the address to the last 16 bits. For example, if the user types in 7777773000G, it would be read as 173000G. Since the memory management unit is disabled by the GO command, the starting address is always in the lower 28 Kwords of memory (0–157776) or the I/O page.

Examples

```
@1000G        ;The program is started at location 1000.

@1000G        ;The program is started with the HALT switch
              ;on. The CPU initializes registers and then
              ;halts without executing the first instruction.

@1000         ;The PC is displayed, then the ODT prompt.
```

3.7.6 P (ASCII 120) Proceed

This command is used to resume execution of a program. The P command corresponds to the CONTINUE switch on other PDP-11 consoles. Program execution resumes at the address pointed to by the PC (R7). The next instruction is fetched and executed; outstanding interrupts are serviced.

If the P command is issued with the front panel switch in the HALT position, it is recognized at the end of instruction execution and ODT is re-entered. The content of the PC (R7) is printed. Using the command, the user can single-instruction step through a program and obtain a PC "trace" on the console terminal.

Examples

```
@R7/002464 1000 Return         ;R7 (PC) is opened and the
                               ;contents displayed. The new
                               ;address is entered in R7.
@P                             ;The proceed command is issued
                               ;and the program continues at
                               ;location 1000.

@P                             ;The proceed command is issued
                               ;with front panel switch in the
001004                         ;Halt position. The PC is
                               ;displayed.
@P                             ;
                               ;Etc.
001010                         ;
@                              ;
```

3.7.7 S Ctrl Shift S Binary Dump

This command is used for test purposes by the manufacturing organization. It is not normally used otherwise. The command is usually received from another computer and not by the console terminal. This command should not be issued from the console terminal because the console ODT echoes back the ASCII 23 code. This may cause the keyboard to lock, thus preventing data from being displayed on the screen.

This command dumps the binary data. Therefore, because the terminal is intended to receive the ASCII data, there is no need to issue this command from a terminal. The command is intended to more efficiently display a portion of the memory as compared to using the slash (/) and Line Feed commands.

This command can be accidentally entered on many terminals by pressing Ctrl S, Ctrl s, or in many cases, by pressing the No Scroll or Hold key. All these conditions normally generate the ASCII 23 code. In order to exit, if the user accidentally enters this command, the user should reset the terminal and type "a" a minimum of three times. This ensures that the console ODT is ready to accept commands again. The command protocol is as follows.

1. After a prompt character, ODT receives a Ctrl S command and echoes it.
2. The host system at the other end of the serial line must send two 8-bit bytes interpreted by ODT as the starting address. These two bytes are not echoed. The first byte specifies bits <15:08>, and the second byte specifies bits <07:00> of the starting address. Bus address bits <21:16> are always forced to 0; the Dump command is restricted to the first 32 Kwords of address space. The starting address may be even or odd.
3. After the second address byte has been received, ODT outputs 10 bytes to the serial line, starting at the address previously specified. When the output is finished, ODT prints Return, Line Feed, @.

Appendix A
CPU Instruction Timing

A.1 Introduction

The execution time for an instruction depends on:

- The type of instruction executed
- The mode of addressing used
- The type of memory being referenced

In general, the total execution time is the sum of the base instruction fetch/execute time plus the operand(s) address calculation/fetch time.

You can use the tables in this section to calculate the length of an instruction in terms of microcycles (MC). Tables A-1 through A-8 list the standard and floating point instructions, their op code listing, and execution times (MC). The "Execution MC" column specifies the number of microcycles required to fetch/execute the base instruction. The R/W column specifies the number of read microcycles (R) and write microcycles (W) in the Execution MC column. Any remaining microcycles are non-I/O (NIO).

If the instruction involves the calculation/fetch of one or more operands, a reference to a separate table (a source or destination table) is made in the last column. The column is usually labeled "Table" or "Dest Table." The tables referenced are A-9, A-10 through A-15, and A-16 through A-20: they are located at the end of the appendix. The source/destination tables specify the number of microcycles the source/destination calculation/fetch requires, and how many of these are read or write microcycles. As before, any remaining microcycles are NIO.

The numbers contained in the tables are based on the assumptions that:

- A memory read must last a minimum of four CLK periods
- A memory write must last a minimum of eight CLK periods
- An NIO lasts four CLK periods (no DMA)

Any wait states caused by slower memory or a DMA transfer must be added to the total instruction time. If wait states are required, the first wait state of a nonstretched read or NIO cycle will last four clock periods and can continue in increments of two clock periods. Further wait states for stretched cycles occur in increments of two clock periods.

Floating-point instruction execution times are given as a range. The actual execution time will vary depending on the type of data being operated on.

The following examples illustrate how to use the tables.

Example 1

How long does a MOV R0,@ 2044 instruction last?

Step 1 From Table A-2, the execution time for the MOV base instruction is 1 MC, or 4 CLK periods. This consists of one read and no write microcycles (R/W column). Depending on the type of memory in the system, the microcycle may be stretched. If so, the microcycle lasts at least 8 CLK periods and may be stretched thereafter in increments of 2 CLK periods.

Step 2 To find the operand calculation/fetch time for the source operand (R0), refer to Table A-9. As shown in Table A-9, a mode 0 register 0 calculate/fetch takes 0 MC. Note that the operand is already available to the DCJ11 (in the register file).

Step 3 To find the operand calculation/fetch time for the destination operand (the contents of memory location 2044), see Table A-12. Table A-12 specifies that a mode 3 register 7 calculate/fetch requires three microcycles (that is, one read microcycle and one write microcycle). Note that the remaining microcycle is an NIO microcycle.

The type of memory in the system must be taken into account. If the read cycle is stretched, the stretched cycle lasts at least 8 CLK periods and may be stretched thereafter in increments of 2 CLK periods. The write microcycle lasts at least 8 CLK periods and may be stretched in increments of 2 CLK periods.

Step 4 For a determination of the minimum time required, total up the microcycles. In this example, it is 1 + 0 + 3, or 4 MC (16 CLK periods if no microcycle stretching occurs).

Example 2

The source and destination tables for floating point instructions show a negative number in the microcycle column for certain mode 2 register 7 operations. For example, to determine how long a CLRD 2000 instruction lasts, you can follow steps 1 through 3:

Step 1 As specified in Table A-8, the base instruction time for the CLRD instruction is 14 MC.

Step 2 From Table A-17, the calculation/fetch time for the operand (a mode 2 register 7 reference) is shown as (-1) under Double Precision. This means that you subtract 1 MC from the base instruction time. However you add 1 MC for the memory write operation. There are no memory read cycles.

Step 3 Total the microcycles:

14 - 1 + 1 = 14 MC minimum.

Note that this example assumes no cycle stretching.

Table A-1 Single Operand Instructions

			Timing			
Mnemonic	Instruction	Op Code Listing	Execution MC	R/W	Source Table	Dest. Table
CLR(B)	Clear	0050DD	1	1/0	-	A-12
COM(B)	Complement (1's)	0051DD	1	1/0	-	A-13
INC(B)	Increment	0052DD	1	1/0	-	A-13
DEC(B)	Decrement	0053DD	1	1/0	-	A-13
NEG(B)	Negate (2's complement)	0054DD	1	1/0	-	A-13
TST(B)	Test	0057DD	1	1/0	-	A-13
Rotate and Shift						
ROR(B)	Rotate right	0060DD	1	1/0	-	A-13
ROL(B)	Rotate left	0061DD	1	1/0	-	A-13
ASR(B)	Arithmetic shift right	0062DD	1	1/0	-	A-13
SWAB	Swap bytes	0003DD	1	1/0	-	A-13
Multiple Precision						
ADC(B)	Add carry	0055DD	1	1/0	-	A-13
SBC(B)	Subtract carry	0056DD	1	1/0	-	A-13
SXT	Sign extend	0067DD	1	1/0	-	A-12
Multiprocessing						
TSTSET	Test and set (low bit interlocked)	0072DD	5	1/1	-	A-13
WRTLCK	Write interlocked	0073DD	4	1/1	-	A-13

Table A-2 Double Operand Instructions

Mnemonic	Instruction	Op Code Listing	Timing: Execution MC	Timing: R/W	Timing: Source Table	Timing: Dest. Table
MOV(B)	Move	01SSDD	1	1/0	A-9	A-13
CMP(B)	Compare	02SSDD	1	1/0	A-9	A-13
ADD	Add	06SSDD	1	1/0	A-9	A-13
SUB	Subtract	16SSDD	1	1/0	A-9	A-13
Logical						
BIT(B)	Bit test (AND)	03SSDD	1	1/0	A-9	A-11
BIC(B)	Bit clear	04SSDD	1	1/0	A-9	A-13
BIS(B)	Bit set (OR)	05SSDD	1	1/0	A-9	A-13
Register						
MUL	Multiply	0704SS	22 (Notes 5, 11)	1/0	-	A-10
DIV	Divide	071RSS	34 (Notes 6, 7, 12)	1/0	-	A-10
ASH	Shift automatically	072RSS	4	1/0	-	A-10
ASHC	Arithmetic shift combined	073RSS	5 (Note 13)	1/0	-	A-10
XOR	Exclusive (OR)	074RDD	1	1/0	-	A-10

Table A-3 Branch Instructions

			Timing			
Mnemonic	Instruction	Branch Op Code Listing	Branch Not Taken MC	Branch Taken R/W	MC	R/W
BR	Branch (unconditional)	000400	2	1/0	4	2/0
BNE	Br if not equal (to 0)	001000	2	1/0	4	2/0
BEQ	Br if equal (to 0)	001400	2	1/0	4	2/0
BPL	Br if plus	100000	2	1/0	4	2/0
BMI	Br if minus	100400	2	1/0	4	2/0
BVC	Br if overflow is clear	102000	2	1/0	4	2/0
BVS	Br if overflow is set	102400	2	1/0	4	2/0
BCC	Br if carry is clear	103000	2	1/0	4	2/0
BCS	Br if carry is set	103400	2	1/0	4	2/0
Signed Conditional Branches						
BGE	Br if greater or equal (to 0)	020000	2	1/0	4	2/0
BLT	Br if less than (0)	002400	2	1/0	4	2/0
BGT	Br if greater than (0)	003000	2	1/0	4	2/0
BLE	Br if less or equal (to 0)	003400	2	1/0	4	2/0

Table A-3 (Cont.) Branch Instructions

Unsigned Conditional Branches						
BHI	Br if higher	101000	2	1/0	4	2/0
BLOS	Br if lower or same	101400	2	1/0	4	2/0
BHIS	Br if higher or same	103000	2	1/0	4	2/0
BLO	Br if lower	103400	2	1/0	4	2/0
SOB	Subtract 1 and branch (if not equal to 0)	077RNN	3	1/0	5	2/0

Table A-4 Jump and Subroutine

			Timing		
Mnemonic	Instruction	Op Code Listing	Execution MC	R/W	Dest. Table
JMP	Jump	0001DD	-	-	A-15
JSR	Jump to subroutine	004RDD	-	-	A-15 (Note 4)
RTS	Return from subroutine	00020R	5	3/0	- (Note 14)
MARK	Stack cleanup	0064NN	10	3/0	

Table A-5 Trap and Interrupt Instructions

			Timing	
Mnemonic	Instruction	Op Code Listing	Execution MC	R/W
EMT	Emulator trap	104000-104377	20	4/2
TRAP	Trap	104400-104777	20	4/2
BPT	Breakpoint trap	000003	20	4/2
IOT	Input/output trap	000004	20	4/2
RTI	Return from interrupt	000002	9	4/0
RTT	Return from interrupt	000006	9	4/0

Table A-6 Condition Code Operators

			Timing	
Mnemonic	Instruction	Op Code Listing	Execution MC	R/W
CLC	Clear C	000241	3	1/0
CLV	Clear V	000242	3	1/0
CLZ	Clear Z	000244	3	1/0
CLN	Clear N	000250	3	1/0
CCC	Clear all CC bits	000257	3	1/0
SEC	Set C	000261	3	1/0
SEV	Set V	000262	3	1/0
SEZ	Set Z	000264	3	1/0
SEN	Set N	000270	3	1/0
SCC	Set all C bits	000277	3	1/0

Table A-7 Miscellaneous Instructions

			Timing		
Mnemonic	Instruction	Op Code Listing	Execution MC	R/W	Dest. Table
HALT	Halt	000000	-		-
WAIT	Wait for interrupt	000001	-		-
RESET	Reset external bus	000005	-		-
NOP	(No operation)	000240	3	1/0	-
SPL	Set priority level to N		7	1/0	-
MFPI	Move from previous instr space	00023N	5	1/1	A-10
MTPI	Move to previous instr space	0056DD	3	2/0	A-12
MFPD	Move from previous data space	1065SS	5	1/1	A-10
MTPD	Move to previous data space	1066DD	3	2/0	A-12
MTPS	Move byte to PSW PS	1064SS	8	1/0	A-10
MFPS	Move byte from PSW PS	1067DD	1	1/0	A-12
MFPT	Move from processor	000007	2	1/0	-
CSM	Call to supervisor mode	0070DD	28	3/3	A-10

Table A-8 Floating-Point Instructions

Mnemonic	Instruction	Op Code Listing	Timing Min	Timing Execution MC Non-Mode 0 Typical	Timing Max	Timing Table
ABSD	Make absolute	1706 fdst	23		24	A-18
ABSF	Make absolute	1706 fdst	19		20	A-18
ADDD	Add	172 (AC) fsvc	41	48	119	A-16
ADDF	Add	172 (AC) fsvc	31	35	102	A-16
CFCC	Copy Floating Condition Codes	170000	5		5	-
CLRD	Clear	1704 fdst	14		14	A-17
CLRF	Clear	1704 fdst	12		12	A-17
CMPD	Compare	173 (AC + 4)	24		25	A-17
CMPF	Compare	173 (AC + 4)	18		19	A-16
DIVD	Divide	174 (AC + 4)	160		167	A-16
DIVF	Divide	174 (AC + 4)	59		63	A-16
LDCDF	Ld & C from D to F	177 (AC + 4)	24		26	A-16
LDCFD	Ld & C from F to D	177 (AC + 4)	20		21	A-16
LDCID	Ld & C Integer to D	177 (AC) src	31		42	A-19
LDCIF	Ld & C Integer to F	177 (AC) src	26		36	A-19
LDCLD	Ld & C Long Integer to D	177 (AC) src	31		42	A-19
LDCLF	Ld & C Long Integer to F	177 (AC) src	26		44	A-19
LDD	Load	172 (AC + 4)	16		17	A-16

Table A-8 (Cont.) Floating-Point Instructions

			Timing			
Mnemonic	Instruction	Op Code Listing	Min	Execution MC Non-Mode 0 Typical	Max	Table
LDEXP	Load Exponent	176 (AC + 4)	17		18	A-19
LDF	Load	172 (AC + 4)	12		13	A-19
LDFPS	Load FPP Program Status	1701 src	6		6	A-19
MODD	Multiply and Separate	171 (AC + 4)	202	217	268	A-16
MODF	Integer and Fraction	171 (AC + 4)	82	94	115	A-16
MULD	Multiply	171 (AC) fsrc	165		173	A-16
MULF	Multiply	171 (AC) fsrc	56		61	A-16
NEGD	Negate	1707 fdst	22		23	A-18
NEGE	Negate	1707 fdst	18		19	A-18
SETD	Set Floating Double Mode	170011	6		6	-
SETF	Set Floating Mode	170001	6		6	-
SETI	Set Integer Mode	170002	6		6	-
SETL	Set Long Integer Mode	170012	6		6	-
STCDF	St & C from D to F	176 (AC) fdst	17		20	A-17
STCDI	St & C from D to Integer	176 (AC) fdst	26		38	A-20
STCDL	St & C from D to Long Integer	176 (AC) fdst	26		54	A-20
STCFD	St & C from F to D	176 (AC) fdst	19		20	A-17
STCFI	St & C from F to Integer	175 (AC + 4)	23		35	A-20

Table A-8 (Cont.) Floating-Point Instructions

			Timing			
Mnemonic	Instruction	Op Code Listing	Min	Execution MC Non-Mode 0 Typical	Max	Table
STCFL	St & C from F to Long Integer	175 (AC + 4)	23		51	A-20
STD	Store	174 (AC) fdst	12		12	A-17
STEXP	Store Exponent	175 (AC) dst	16		16	A-20
STF	Store	174 (AC) fdst	8		8	A-17
STFPD	Store FPP Program Status	1702 dst	9		9	A-20
STST	Store FPP Status	1703 dst	7		7	A-20
SUBD	Subtract	173 (AC) fsrc	47	55	122	A-16
SUBF	Subtract	173 (AC) fsrc	37	41	104	A-16
TSTD	Test	1705 fdst	11		12	A-16
TSTF	Test	1705 fdst	9		10	A-16

Table A-9 Source Address Times: All Double Operand

Source Mode	Source Register	Microcode Cycles	Read Memory Cycles
0	0-7	0	0
1	0-7	2	1
2	0-6	2	1
2	7	1	1
3	0-6	4	2
3	7	3	2
4	0-6	3	1
4	7	6	2 (Note 1)
5	0-6	5	2
5	7	8	3 (Note 1)
6	0-7	4	2
7	0-7	6	3

Table A-10 Destination Address: Read-Only Single Operand

Destination Mode	Destination Register	Microcode Cycles	Read Memory Cycles
0	0-7	0	0
1	0-7	2	1
2	0-6	2	1
2	7	1	1
3	0-6	4	2
3	7	3	2
4	0-6	3	
4	7	7	2 (Note 2)
5	0-6	5	2
5	7	9	3 (Note 3)
6	0-7	4	2
7	0-7	6	3

Table A-11 Destination Address Times: Read-Only Double Operand

Destination Mode	Destination Register	Microcode Cycles	Read Memory Cycles
0	0-7	0	0
1	0-7	3	1
2	0-6	3	1
2	7	2	1
3	0-6	5	2
3	7	3	2
4	0-6	4	1
4	7	8	2 (Note 2)
5	0-6	6	2
5	7	10	3 (Note 3)
6	0-7	5	2
7	0-7	7	3

Table A-12 Destination Address Times: Write-Only

Destination Mode	Destination Register	Microcode Cycles	Memory Read	Cycles Write
0	0-6	0	0	0
0	7	5	1	0
1	0-6	2	0	1
1	7	6	1	1
2	0-6	2	0	1
2	7	6	1	1
3	0-6	4	1	1
3	7	3	1	1
4	0-6	3	0	1
4	7	7	1	1
5	0-6	5	1	1
5	7	9	2	1
6	0-7	4	1	1
7	0-7	6	2	1

Table A-13 Destination Address Times: Read Modify Write

Destination Mode	Destination Register	Microcode Cycles	Memory Read	Cycles Write
0	0-6	0	0	0
0	7	5	1	0
1	0-6	3	1	
1	7	7	2	1
2	0-6	3	1	1
2	7	7	2	1
3	0-6	5	2	1
3	7	4	2	1
4	0-6	4	1	1
4	7	8	2	1 (Note 2)
5	0-6	6	2	1
5	7	10	3	1 (Note 3)
6	0-7	5	2	1
7	0-7	7	3	1

Table A-14 Destination Address Times: JMP

Destination Mode	Destination Register	Microcode Cycles	Memory Read	Cycles Write
1	0-7	4	2	0
2	0-7	6	2	0
3	0-7	5	3	0
4	0-7	5	2	0
5	0-7	6	3	0
6	0-6	6	3	0
6	7	5	3	0
7	0-7	7	4	0

Table A-15 Destination Address Times: JSR

Destination Mode	Destination Register	Microcode Cycles	Memory Read	Cycles Write
1	0-7	9	2	1
2	0-7	10	2	1
3	0-6	10	3	1
3	7	9	3	1
4	0-7	10	2	1
5	0-7	11	3	1
6	0-6	10	3	1
6	7	9	3	1
7	0-7	12	4	1

Table A–16 Floating Source 1-7

Microcode Mode	Memory Register	Memory Cycles	Read	Write
Single Precision				
1	0-7	3	2	0
2	0-6	3	2	0
2	7	1	1	0
3	0-6	4	3	0
3	7	3	3	0
4	0-7	4	2	0
5	0-7	5	3	0
6	0-7	4	3	0
7	0-7	6	4	0
Double Precision				
1	0-7	5	4	0
2	0-6	5	4	0
2	7	0 (Note 15)	1	0
3	0-6	6	5	0
3	7	5	5	0
4	0-7	6	4	0
5	0-7	7	5	0
6	0-7	6	5	0
7	0-7	8	6	0

Table A-17 Floating Destination Modes 1-7

Microcode Mode	Memory Register	Memory Cycles	Read	Write
Single Precision				
1	0-7	3	0	2
2	0-6	3	0	2
2	7	1	0	1
3	0-6	4	1	2
3	7	3	1	2
4	0-7	4	0	2
5	0-7	5	1	2
6	0-7	4	1	2
7	0-7	6	2	2
Double Precision				
1	0-7	5	0	4
2	0-6	5	0	4
2	7	(-1) (Note 15)	0	1
3	0-6	6	1	4
3	7	5	1	4
4	0-7	6	0	4
5	0-7	7	1	4
6	0-7	6	1	4
7	0-7	8	2	4

Table A–18 Floating Read-Modify-Write Modes 1-7

Microcode Mode	Memory Register	Memory Cycles	Read	Write
Single Precision				
1	0-7	5	2	2
2	0-6	5	2	2
2	7	1 (Note 15)	1	1
3	0-6	6	3	2
3	7	5	3	2
4	0-7	6	2	2
5	0-7	7	3	2
6	0-7	6	3	2
7	0-7	8	4	2
Double Precision				
1	0-7	9	4	4
2	0-6	9	4	4
2	7	(-2) (Note 15)	1	1
3	0-6	10	5	4
3	7	9	5	4
4	0-7	10	4	4
5	0-7	11	5	4
6	0-7	10	5	4
7	0-7	12	6	4

Table A-19 Integer Source Modes 1-7

Microcode Mode	Memory Register	Memory Cycles	Read	Write
Integer				
1	0-7	2	1	0
2	0-6	2	1	0
2	7	0 (Note 15)	1	0
3	0-6	3	2	0
3	7	2	2	0
4	0-7	3	1	0
5	0-7	4	2	0
6	0-7	3	2	0
7	0-7	5	3	0
Long Integer				
1	0-7	4	2	0
2	0-6	4	2	0
2	7	0 (Note 15)	1	0
3	0-6	5	3	0
3	7	4	3	0
4	0-7	5	2	0
5	0-7	6	3	0
6	0-7	5	3	0
7	0-7	7	4	0

Table A-20 Integer Destination Modes 1-7

Microcode Mode	Memory Register	Memory Cycles	Read	Write
Integer				
1	0-7	2	0	1
2	0-6	2	0	1
2	7	2	0	1
3	0-6	3	1	1
3	7	2	1	1
4	0-7	3	0	1
5	0-7	4	1	1
6	0-7	3	1	1
7	0-7	5	2	1
Long Integer				
1	0-7	4	0	2
2	0-6	4	0	2
2	7	2	0	1
3	0-6	5	1	2
3	7	4	1	2
4	0-7	5	0	2
5	0-7	6	1	2
6	0-7	5	1	2
7	0-7	7	2	2

A.2 Source and Destination Table Notes

1. Subtract 2 MC and 1 read if both source and destination modes autodecrement PC, or if write-only or read-modify-write mode 07 or 17 is used.
2. Read-only and read-modify-write destination mode 47 references actually perform 3 READ operations. For bookkeeping purposes, one of the reads is accounted for in the execute, fetch timing.
3. READ-ONLY and READ-MODIFY-WRITE destination mode 57 references actually perform 4 READ operations. For bookkeeping purposes, one of the READs is accounted for in the EXECUTE, FETCHING TIMING.
4. Subtract 1 MC if the link register is PC.
5. Add 1 MC if the source operand is negative.
6. Subtract 1 MC if the source mode is not 0.
   a. Add 1 MC if the quotient is even.
   b. Add 2 MC if overflow occurs.
   c. Add 5 MC and 1 read if the PC is used as a destination register, but only if source mode 47 or 57 is not used.
7. Add 1 MC per shift.

8. Add 1 MC if source operand <15:6> is not 0.
9. Subtract 1 MC if one shift only.
10. Add 4 MC and 1 read if the PC is used as a destination register, but only if source mode 47 or 57 is not used.
11. Divide by zero executes in 5 MC (see Note 6).
12. Timing for no shift. Add 1 MC if a left shift. (Notes 8, 9, 11 apply.) Add 2 MC for a right shift. (Notes 8, 10, 11 apply.)
13. Add 1 MC if a register other than R7 is used.
14. Mode 27 references only access single-word operands. The execution time listed has been compensated in order to computer the total execution time accurately.

Appendix B
PDP-11/84 Hardware/Software Differences

B.1 UNIBUS Power-Up Protocol Differences

The UNIBUS power-up protocol on PDP-11/84 systems is different than on most PDP-11 systems. (See Figure B-1.) On most PDP-11 systems, the UNIBUS signal INITIALIZE (INIT) L is held asserted for a minimum of 10 milliseconds after the negation of DC LINE LOW (DC LO L) on power-up. On PDP-11/84 systems the UNIBUS signal INIT L is held asserted for a minimum of 16 microseconds after the negation of DC LO L on power-up. This difference will not affect system operations.

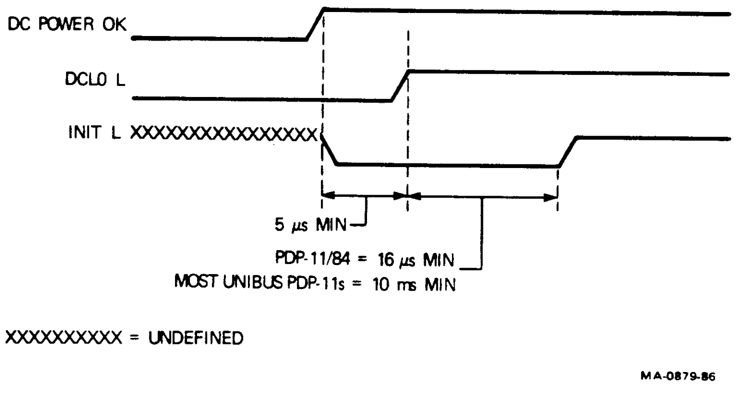

Figure B-1 Power-Up Protocol Timing Differences

B.2 PDP-11/84 and PDP-11/44 Hardware Differences

Products based on the PDP-11/84 may replace products based on the PDP-11/44 for certain applications. However, PDP-11/84-based products do not contain the following PDP-11/44 hardware features.

- Cache data register (17777754)
- Switch register (17777570)

Products based on the PDP-11/84 contain the following functionality that is not present in PDP-11/44 products.

- Dual general register set
- SPL, MTPS, MFPS, TSTSET, WRTLCK instructions

The following registers are not implemented on the PDP-11/44. These registers are used primarily for testing by the CPU ROM code, other diagnostics, or for system configuration by the CPU ROM code. These registers are not normally used by system-level software.

- Boot and diagnostic controller register (17777520)
- ROM page control register (17777522)
- Configuration and display register (17777524)
- Diagnostic controller register (17777730)
- Diagnostic data register (17777732)
- Memory configuration register (17777734)

DMA transfers cannot occur between UNIBUS peripherals and any registers on the CPU. DMA transfers can only occur between the UNIBUS peripherals and the UBA. DMA transfers may also occur between UNIBUS peripherals and the addresses of the ROM sockets on the UBA (17 773 000–17 773 776).

Table B–1 summarizes the hardware differences between products based on the PDP-11/44 and products based on the PDP-11/84.

Table B-1 Hardware Differences Between the PDP-11/84 and PDP-11/44

Address	Function	Differences
17777776	PS	Added register set select bit <11>
17777772	PIRQ	No diffference
17777766	CPU error	UNIBUS monitoring bits not implemented
17777754	Cache data	Not implemented
17777752	Hit/miss	No difference
17777750	Maintenance	Hardware differences (see Chapter 2)
17777746	Cache control	Hardware differences (see Chapter 2)
17777744	Memory error	Hardware differences (see Chapter 2)
17777676 to 17777660	User data PAR	No difference
17777656 to 17777640	User instruction PAR	No difference
17777636 to 17777620	User data PDR	No difference
17777616 to 17777600	User instruction PDR	No difference
17777576	MMR2	No difference
17777574	MMR1	No difference
17777572	MMR0	Elminated maintenance mode
17777570	Switch register	Not implemented
17772516	MMR3	No difference
17772376 to 17772360	Kernel data PAR	No difference
17772356 to 17772340	Kernel instruction PAR	No difference
17772336 to 17772320	Kernel data PDR	No difference

Table B-1 (Cont.) Hardware Differences Between the PDP-11/84 and PDP-11/44

Address	Function	Differences
17772316 to 17772300	Kernel instruction	No difference
17772276 to 17772260	Supervisor data PAR	No difference
17772256 to 17772240	Supervisor instruction PAR	No difference
17772236 to 17772220	Supervisor data PDR	No difference
17772216 to 17772200	Supervisor instruction PDR	No difference

B.3 PDP-11/84 and PDP-11/70 Hardware Differences

The PDP-11/84 may replace the PDP-11/70 in certain applications. However, the PDP-11/84 does not have the following PDP-11/70 hardware features.

- Stack limit register (17777774)
- Micro break register (17777770)
- System ID register (17777764)
- System size registers (17777760, 17777762)
- Physical error address registers (17777740, 17777742)
- Switch register (17777570)

Products based on the PDP-11/84 contain the following functionality that is not present in PDP-11/70 products.

- MTPS, MFPS, MFPT, CSM, TSTSET, WRTLCK instructions
- Bypass cache bit in PDRs

The following registers are not implemented on the PDP-11/44. These registers are used primarily for testing by the CPU ROM code, for other diagnostics, or for system configuration by the CPU ROM code. These registers are not normally used by system-level software.

- Boot and diagnostic controller register (17777520)
- ROM page control register (17777522)
- Configuration and display register (17777524)
- Diagnostic controller register (17777730)

- Diagnostic data register (17777732)
- Memory configuration register (17777734)

DMA transfers cannot occur between UNIBUS peripherals and any registers on the CPU. DMA transfers can only occur between the UNIBUS peripherals and the UBA. DMA transfers may also occur betwen UNIBUS peripherals and the addresses of the ROM sockets on the UBA (17 773 000-17 773 776).

Table B-2 summarizes the hardware differences between products based on the PDP-11/70 and products based on the PDP-11/84.

Table B-2 Hardware Differences Between the PDP-11/84 and PDP-11/70

Address	Function	Differences
17777776	PS	Added suspended instruction bit <8>
17777774	Stack limit	Not implemented
17777772	PIRQ	No diffference
17777770	Micro break	Not implemented
17777766	CPU error	No difference
17777764	System ID	Not implemented
17777760	System size	No difference
17777752	Hit/miss	No difference
17777750	Maintenance	Hardware differences (see Chapter 2)
17777746	Cache control	Hardware differences (see Chapter 2)
17777744	Memory error	Hardware differences (see Chapter 2)
17777742	High error address	Not implemented
17777740	Low error address	Not implemented
17777676 to 17777660	User data PAR	No difference
17777656 to 17777640	User instruction PAR	No difference
17777636 to 17777620	User data PDR	Added bypass cache, eliminated access flags and access modes other than 0, 2, and 6
17777616 to 17777600	User instruction PDR	Added bypass cache, eliminated access flags and access modes other than 0, 2, and 6

Table B-2 (Cont.) Hardware Differences Between the PDP-11/84 and PDP-11/70

Address	Function	Differences
17777574	MMR1	No difference
17777572	MMR0	Elminated traps, maintenance mode, and instruction complete
17777570	Switch register	Not implemented
17772516	MMR3	Added CSM enable bit <3>
17772376 to 17772360	Kernel data PAR	No difference
17772356 to 17772340	Kernel instruction PAR	No difference
17772336 to 17772320	Kernel data PDR	Added bypass cache, eliminated access flag and access mode other than 0, 2, and 6
17772316 to 17772300	Kernel instruction PDR	Added bypass cache, eliminated access flag and access modes other than 0, 2, and 6
17772276 to 17772260	Supervisor data PAR	No difference
17772256 to 17772240	Supervisor instruction PAR	No difference
17772236 to 17772220	Supervisor data PDR	Added bypass cache, eliminated access flag and access modes other than 0, 2, and 6
17772216 to 17772200	Supervisor instruction PDR	Added bypass cache, eliminated access flag and access modes other than 0, 2, and 6

B.4 Software Differences

Table B-3 summarizes the programming differences, at the assembly language level, between the DCJ11 and other processors in the PDP-11 family.

Table B-3 Programming Difference for PDP-11 Family Processors

Item	Processors 23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
1. OPR %R, (R) +; OPR %R, - (R) using the same register as both source and destination: contents of R are incremented (decremented) by 2 before being used as the source operand.	x						x	x			x	x	x	
OPR %R, (R) +: OPR %R, - (R) using the same register as both register and destination: initial contents of R are used as the source operand.		x	x	x	x	x			x	x				x
2. OPR %R, @(R) +; OPR %R, @ - (R) using the same register as both source and destination: contents of R are incremented (decremented) by 2 before being used as the source operand.	x						x	x			x	x	x	
OPR %R, @ (R) +; OPR %R, @ - (R) using the same register as both source and destination: initial contents of R are used as the source operand.		x	x	x	x	x			x	x				x
3. OPR PC, X (R); OPR PC, @ X (R); OPR PC, @ A; location A will contain the PC of OPR +4.	x						x	x			x	x	x	
OPR PC, X (R); OPR PC, @ X (R), OPR PC, A; OPR PC, @ A: location A will contain the PC of OPR +2.		x	x	x	x	x			x	x				x
4. JMP (R) + or JSR reg, (R) +: contents of R are incremented by 2, then used as the new PC address.						x	x							
JMP (R) + or JSR reg, (R) +: initial contents of R are used as the new PC.	x	x	x	x	x			x	x	x	x	x	x	x

Table B-3 (Cont.) Programming Difference for PDP-11 Family Processors

Item	Processors 23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
5. JMP %R or JSR reg. %R traps to 10 (illegal instruction).	X		X	X	X	X	X	X			X		X	NA
JMP %R or JSR reg. %R traps to 4 (illegal instruction).		X							X	X		X		NA
6. SWAB does not change V.							X							
SWAB clears V.	X	X	X	X	X	X		X	X	X	X	X	X	X
7. Register addresses (177700-177717) are valid program addresses when used by CPU.						X							1	1
Register addresses (177700-177717) time out when used as a program address by the CPU. Can be addressed under console operation.		X	X	X			X	X	X	X	X			NA
Register addresses (177700-177717) time out when used as an address by CPU or console.	X				X							X		
8. Basic instructions noted in PDP-11 processor handbook.	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SOB. MAR. RTT. SXT instructions[2]	X	X		X	X			X	X	X	X	X	X	3
ASH. ASHC. DIV. MUL. XOR	X	X		X	X			X	X	X	X	X		X
Floating Point Instructions in base machine.											X	X		
MFPT Instruction	X	X										X		
The external option KE11-A provides MUL. DIV. SHIFT operation in the same data format.						X	X							
The KE11-E (Expansion Instruction Set) provides the instructions MUL. DIV. ASH. and ASHC. These new instructions are 11/45 compatible.								X						

[1]Register addresses (177700-177717) are handled as regular memory addresses in the I/O page.

[2]RTT instruction is available in 11/04 but is different than other implementations.

[3]All but MARK.

Table B-3 (Cont.) Programming Difference for PDP-11 Family Processors

	Processors													
Item	23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
The KE11-F (Floating Instruction Set) adds unique stack ordered oriented point instructions: FADD. FSUB. FMUL. FDIV.								X						
The KEV-11 adds EIS/FIS instructions					X									
MFP. MTP instructions	X	X		X				X		X	X	X		
SPL instruction		X							X	X		X		
CSM instruction		X										X		
9. Power fall during RESET instruction is not recognized until after the instruction is finished (70 milliseconds). RESET instruction consists of 70 millisecond pause with INIT occurring during first 20 milliseconds.							X	X			X			
Power fail immediately ends the RESET instruction and traps if an INIT is in progress. A minimum INIT of 1 microsecond occurs if instruction aborted. PDP-11-04/34/44 are similar with no minimum INIT time.		X	X	X					X	X				
Power fail acts the same as 11/45 (22 milliseconds with about 300 nanoseconds minimum). Power fail during RESET fetch is fatal with no power-down sequence.						X								
RESET instruction consists of 10 microseconds of INIT followed by a 90 microsecond pause. Reset instruction consists of a minimum 8.4 microseconds followed by a minimum 100 nanosecond pause. Power fail not recognized until the instruction completes.	X				X							X		
10. No RTT instruction						X	X							
If RTT sets the "T" bit. the "T" bit trap occurs after the instruction following RTT.	X	X	X	X	X			X	X	X	X	X	X	X

Table B-3 (Cont.) Programming Difference for PDP-11 Family Processors

	Processors													
Item	23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
11. If RT1 sets "T" bit, "T" bit trap is acknowledged after instruction following RT1.						X	X							X
If RT1 sets "T" bit, "T" bit trap is acknowledged immediately following RT1.	X	X	X	X	X			X	X	X	X	X	X	
12. If an interrupt occurs during an instruction that has the "T" bit set, the "T" bit trap is acknowledged before the interrupt.	X	X	X	X	X	X	X	X			X	X	X	4
If an interrupt occurs during an instruction and the "T" bit is set, the interrupt is acknowledged before "T" bit trap.									X	X				NA
13. "T" bit trap will sequence out of WAIT instruction.	X	X	X	X		X	X	X			X		X	NA
"T" bit trap will not sequence out of WAIT instruction. Waits until an interrupt.					X				X	X				
14. Explicit reference (direct access) to PS can load "T" bit. Console can also load "T" bit.			X			X	X							
Only implicit references (RT1, RTT, traps and interrupts) can load "T" bit. Console cannot load "T" bit.	X	X		X	X			X	X	X	X	X	X	X
15. Odd address/nonexistent references using the SP cause a HALT. This is a case of double bus error with the second error occurring in the trap servicing the first error. Odd address trap not implemented in LS1-11, 11/23 or 11/24.		X	X	X	X	X	X							
Odd address/nonexistent references using the stack pointer cause a fatal trap. On bus error in trap service, new stack created at 0/2.	X							X	X	X	X	X	5	6

[4]Interrupts not visible to VAX compatibility mode.

[5]Odd address/nonexistent references using SP do not trap.

[6]Odd address aborts to native mode.

Table B-3 (Cont.) Programming Difference for PDP-11 Family Processors

Item	Processors 23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
16. The first instruction in an interrupt routine will not be executed if another interrupt occurs at a higher priority level than assumed by the first interrupt.	X	X	X	X	X	X		X	X	X	X	X	X	X
The first interrupt in an interrupt service is guaranteed to be executed.							X							
17. Single general purpose register set implemented.	X	X	X	X	X	X	X	X			X		X	X
Dual general purpose register set implemented.									X	X		X		
18. PSW address. 177776. not implemented; must use instructions MTPS (move to PS) and MFPS (move from PS).					X								X	7
PSW address implemented. MTPS and MFPS not implemented.		X	X			X	X	X	X	X	X			
PSW address and MTPS and MFPS implemented.	X			X								X		
19. Only one interrupt level (BR4) exists.					X									
Four interrupt levels exist.	X	X	X	X		X	X	X	X	X	X	X	X	NA
20. Stack overflow not implemented.					X								X	X
Some sort of stack overflow implemented.	X	X	X	X		X	X	X	X	X	X	X		
21. Odd address trap not implemented.	X				X								X	
Odd address trap implemented.		X	X	X		X	X	X	X	X	X	X		X
22. FMUL and FDIV instructions implicitly use R6 (one push and pop); hence R6 must be set up correctly.				X										
FMUL and FDIV instructions do not implicitly use R6.								X						NA

[7]Can reference PSW only from native mode.

Table B-3 (Cont.) Programming Difference for PDP-11 Family Processors

Item	Processors 23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
23. Due to their execution time. EIS instructions can abort because of a device interrupt.					X									X
EIS instructions do not abort because of a device interrupt.	X	X		X				X	X	X	X	X		NA
24. Due to their execution time. FIS instructions can abort because of a device interrupt.					X			X						NA
25. Due to their execution time. FP11 instructions can abort because of a device interrupt.[8]	X													
FP11 instructions do not abort because of a device interrupt.		X		X					X	X	X	X		NA
26. EIS instructions do a DATIP and DATO bus sequence when fetching source operand.					X									
EIS instructions do a DATI bus sequence when fetching source operand.	X	X		X				X	X	X	X	X		NA
27. MOV instruction does just a DATO bus sequence for the last memory cycle.	X	X		X	X			X	X	X	X	X		9
MOV instruction does a DATIP and DATO bus sequence for the last memory cycle.			X			X	X						10	
28. If PC contains nonexistent memory and a bus error occurs. PC will have been incremented.	X	X	X	X	X	X	X		X	X		X		
If PC contains nonexistent memory address and a bus error occurs. PC will be unchanged.								X					11	X
29. If register contains nonexistent memory address in mode 2 and a bus error occurs. register will be incremented.	X				X	X	X	X	X	X		X	11	

[8]Integral floating point assumed on 11/23 and 11/24, FP11E assumed on 11/60.

[9]Implementation dependent.

[10]MOV instruction does a DATI and a DATO bus sequence for last memory cycle.

[11]Does not support bus errors.

Table B-3 (Cont.) Programming Difference for PDP-11 Family Processors

Item	Processors 23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
Same as above but register is unchanged.		X	X	X										
30. If register contains an odd value in mode 2 and a bus error occurs, register will be incremented.	X				X			X	X	X		X	11	12
If register contains an odd value in mode 2 and a bus error occurs, register will be unchanged.		X	X	X		X	X							
31. Condition codes restored to original values after FIS interrupt abort (EIS doesn't abort on 35/40).							X							
Condition codes that are restored after EIS/FIS interrupt abort are indeterminate.					X									NA
32. Opcodes 075040 through 075377 unconditionally trap to 10 as reserved opcodes.	X	X	X	X	X	X	X	X	X	X	X	X	X	13
If KEV-11 option is present, opcodes 75040 through 07533 perform a memory read using the register specified by the low order 3 bits as a pointer. If the register contents is a nonexistent address, a trap to 4 occurs. If the register contents is an existent address, a trap to 10 occurs.					X									
33. Opcodes 210 through 217 trap to 10 as reserved instructions.	X	X	X	X		X	X	X	X	X	X	X	X	13
Opcodes 210 through 217 are used as a maintenance instruction.					X									
34. Opcodes 75040 through 75777 trap to 10 as reserved instructions.	X	X	X	X		X	X	X	X	X	X	X	X	11

[11]Does not support bus errors.

[12]Unpredictable.

[13]Traps to native mode.

Table B-3 (Cont.) Programming Difference for PDP-11 Family Processors

	Processors													
Item	23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
If KEV-11 option is present, opcodes 75040 through 75577 can be used as escapes to user microcode. If no user microcode exists, a trap to 10 occurs.					X									
35. Opcodes 170000 through 177777 trap to 10 as reserved instructions.			X			X	X	X					X	[11]
Opcodes 170000 through 177777 are implemented as floating point instructions.	X	X		X					X	X	X	X		
Opcodes 170000 through 177777 can be used as escapes to user microcode. If no user microcode exists, a trap to 10 occurs.					X									
Opcode 076600 used for maintenance.														
36. CLR and SXT do just a DATO sequence for the last bus cycle.	X											X		[12]
CLR and SXT do DATIP-DATO sequence for the last bus cycle.		X	X	X	X	X	X	X	X	X	X		[14]	
37. MEM MGT maintenance mode MMRO bit 8 is implemented.		X		X				X	X	X	X			
MEM MGT maintenance mode MMRO bit 8 is not implemented.	X											X		NA
38. PS<15:12>, nonkernel mode, nonkernel stack pointer and MTPx and MFPx instructions exist even when MEM MGT is not configured.	X	X							X	X	X	X		
PS<15:12>, nonkernel mode, nonkernel stack pointer, and MTPx and MFPx instructions exist only when MEM MGT is configured.								X						NA

[11]Does not support bus errors.

[12]Unpredictable.

[14]CLR and SXT do DATI-DATO.

Table B-3 (Cont.) Programming Difference for PDP-11 Family Processors

Item	Processors 23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
39. Current mode PS bits <15:14> set to 01 or 10 will cause a MEM MGT trap upon any memory reference.	X			X				X						
Current mode PS bits <15:14> set to 10 will be treated as kernel mode (00) and not cause a MEM MGT trap.	X													NA
Current mode PS bits <15:14> set to 10 will cause a MEM MGT trap upon any memory reference.		X							X	X		X		
40. MTPS in user mode will cause MEM MGT trap if PS address 177776 not mapped. If mapped. PS <7:5> and <3:0> affected.				X										
MTPS in nonuser mode will not cause MEM MGT trap and will only affect PS <3:0> regardless of whether PS address 177776 is mapped.	X											X		NA
41. MFPS in user mode will cause MEM MGT if PS address 177776 not mapped. If mapped. PS <7:0> are accessed.				X										
MTPS in user mode will not trap regardless of whether PS address 177776 is mapped.	X											X		NA
42. Programs cannot execute out of internal processor registers.												X		
Programs can execute out of internal processor registers.	X	X		X				X	X	X	X			
43. A HALT instruction in user or supervisor mode will trap through location 4.		X							X	X		X		
A HALT instruction in user or supervisor mode will trap through location 10.	X			X				X			X		[13]	[15]

[13]Traps to native mode.

[15]HALT pushes PC and PSW to stack, loads PS with 340 and PC with <powerup address> +40.

Table B-3 (Cont.) Programming Difference for PDP-11 Family Processors

Item	Processors 23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
44. PDR bit <0> implemented.									X	X				
PDR bit <0> not implemented.	X	X		X				X			X	X	X	X
45. PDR bit <7> (any access) implemented.									X	X				
PDR bit <7> (any access) not implemented.	X	X		X				X			X	X	X	X
46. Full PAR <15:0> implemented.	X	X								X		X		
Only PAR <11:0> implemented.				X				X	X		X		X	X
47. MMR0 <12> - trap-memory management implemented.									X	X				
MMR0 <12> not implemented.	X	X		X				X			X	X	X	X
48. MMR3 <2:0> - D space enable implemented.		X							X	X		X		
MMR3 <2:0> not implemented.	X			X				X			X		X	X
49. MMR3 <5:4> - IOMAP, 22-bit mapping enabled implemented.	X	X								X		X		
MMR3 <5:4> not implemented.				X				X	X		X		X	X
50. MMR3 <3> - CSM enable implemented.		X										X		
MMR3 <3> not implemented.	X			X				X	X	X	X		X	X
51. MMR2 tracks instruction fetches and interrupt vectors.									X	X				
MMR2 tracks only instruction fetches.	X	X		X				X			X	X	NA	NA

Table B-3 (Cont.) Programming Difference for PDP-11 Family Processors

Item	Processors 23/24	44	04	34	LSI-11	05/10	15/20	35/40	45	70	60	J-11	T-11	VAX
52. MFPx %6, MTPx when PS <13:12> = 10 gives unpredictable results.	X	X		X				X	X	X	X			
MTPx %6, MTPx %6 when PS <13:12> = 10 uses user stack pointer.												X	NA	NA
55. The ASH instruction with a source operand of octal 37 (shift left 31 decimal times) causes the register to be shifted right instead of left.												X		
56. The ASHC instruction with an octal value of 37 (shift left 31 decimal times) in source operand bits <5:0> and bits <15:6> of the operand being nonzero, causes the register to be shifted right instead of left.												X		

Appendix C
Configuration Register Modification

C.1 Introduction

Figure C–1 shows the format of the boot and diagnostic configuration register (BCR). It also shows the connections to external switches that allow the register values to be modified without gaining access to the CPU module. Since bits <15:8> of the register are not driven, they may be read as 1 or 0.

In order for a register bit value to be remotely controlled, the switch that controls that bit on the KDJ11-BF module must be off to allow control to be passed to the external switch. When a KDJ11-BF module switch is on, the corresponding line is grounded. This causes the register bit to always be read as a 0 regardless of the position of any external switch.

Bits <7:4> (switches 1-4) are not connected remotely on PDP-11/84 systems. The bits must be set at the KDJ11-BF module switchpack unless a custom cable is built and connected to an external switch by the user.

The eight-position switchpack is mounted at the handle end of the KDJ11-BF. Switches 6-8 select the baud rate for the DLART chip. Switches 1-8 correspond to bits <07:00> in the read-only BCR at address 17777524.

C.2 ROM Code Interpretation

The following paragraphs describe how the ROM code interprets the values of the BCR when it is read. The ROM code only looks at register bits <7:3> (switches 1-5).

NOTE

The following discussion assumes that when a switch is off, the connection to it is pulled high by the CPU module. If an external device is grounding a switch line, then the switch is considered on.

When using an external device to control the selections, any switch on the CPU that goes to an external switch should be off in order to pass control to the external device.

Normally switches 1-5 are off, and the EEPROM contents determine what action is to be taken at power-up or restart.

When off, switch 5 unconditionally forces the ROM code to enter dialog mode at the completion of the selected tests. Autoboot cannot occur if switch 5 is off.

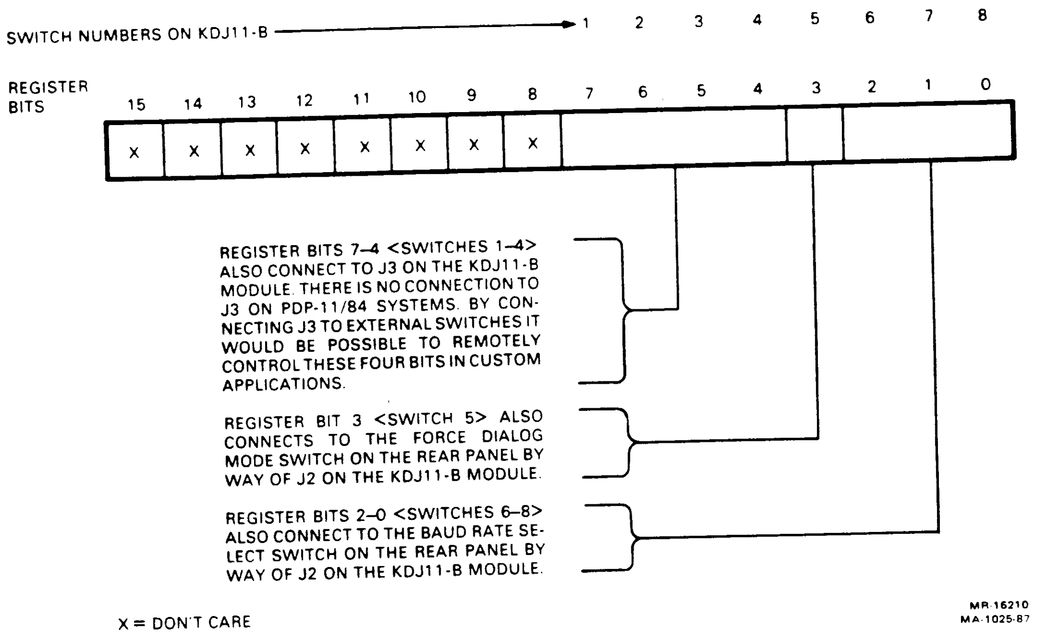
SWITCH NUMBERS ON KDJ11-B
1 2 3 4 5 6 7 8
REGISTER BITS
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
X X X X X X X X
REGISTER BITS 7–4 <SWITCHES 1–4> ALSO CONNECT TO J3 ON THE KDJ11-B MODULE. THERE IS NO CONNECTION TO J3 ON PDP-11/84 SYSTEMS. BY CONNECTING J3 TO EXTERNAL SWITCHES IT WOULD BE POSSIBLE TO REMOTELY CONTROL THESE FOUR BITS IN CUSTOM APPLICATIONS.
REGISTER BIT 3 <SWITCH 5> ALSO CONNECTS TO THE FORCE DIALOG MODE SWITCH ON THE REAR PANEL BY WAY OF J2 ON THE KDJ11-B MODULE.
REGISTER BITS 2–0 <SWITCHES 6–8> ALSO CONNECT TO THE BAUD RATE SELECT SWITCH ON THE REAR PANEL BY WAY OF J2 ON THE KDJ11-B MODULE.
X = DON'T CARE
MR-16210
MA-1025-87

Figure C–1 Boot and Diagnostic Configuration Register

If switches 5 and 1 are on, the console is disabled and all output to the console is suppressed. Dialog mode cannot be entered. If any input occurs at the console, the program will transmit an error message to the console indicating the console is disabled.

If switch 5 is on and switches 2-4 are not equal to octal 0 or 7, then autoboot mode is selected. The device and unit number to be booted are determined by a table in the EEPROM and the value in switches 2-4 (1-6).

There are default values in the table for the PDP-11/84 system. If these default devices do not meet the needs of the user, they can be changed to any value in setup mode. The selections can be standard ROM boots, UBA ROM boots, or any EEPROM boots. The selections are changed using SETUP command 6.

See Paragraph 3.3.4 for more information on Setup command 4.

See Paragraph 3.3.6 for more information on Setup command 6.

```
Interpretation of the BCR by the ROM code.

1 = OFF          0 = ON         X = Don't care

1 2 3 4 5 6 7 8      Switch number on KDJ11-B module
7 6 5 4 3 2 1 0      Register bit

                     OCTAL     ACTION TAKEN AT POWER UP

X X X X 1 X X X                Console enabled. Unconditionally
                               transfer control to dialog after
                               running tests. (FORCED DIALOG)
```

The forced dialog switch—located at the rear of the cabinet or box - drives bit 3 low (0) when it is disabled. When the forced dialog switch is enabled, bit 3 is high (1) as long as switch 5 on the KDJ11-B module is off.

In the following example, the console is enabled. Switches 6-8 select the baud rate. Autoboot mode is automatically selected for selections switch boot 1 to switch boot 6 (SB n) from setup mode command 6.

1 1 1 1 0 X X X	36X	Dispatch according to EEPROM
1 1 1 0 0 X X X	34X	Autoboot 6th selection from SETUP command 6
1 1 0 1 0 X X X	32X	Autoboot 5th selection from SETUP command 6
1 1 0 0 0 X X X	30X	Autoboot 4th selection from SETUP command 6
1 0 1 1 0 X X X	26X	Autoboot 3rd selection from SETUP command 6
1 0 1 0 0 X X X	24X	Autoboot 2nd selection from SETUP command 6
1 0 0 1 0 X X X	22X	Autoboot 1st selection from SETUP command 6
1 0 0 0 0 X X X	20X	Power up to ODT immediately

Dispatch according to EEPROM has four possible modes:

- Dialog mode
- Autoboot mode, trying one to six devices defined by setup mode command 4
- Halt and enter ODT
- Vector through location 24/26 (power fail recovery)

For the following example, the console is disabled and switches 6-8 are not used. Autoboot mode is automatically selected.

```
1 = OFF          0 = ON          X = Don't care

1 2 3 4 5 6 7 8     Switch number on KDJ11-B module
7 6 5 4 3 2 1 0     Register bit
```

	OCTAL	ACTION TAKEN AT POWER UP
0 1 1 1 0 X X X	16X	Autoboot according to SETUP command 4
0 1 1 0 0 X X X	14X	Autoboot 6th selection from SETUP command 6
0 1 0 1 0 X X X	12X	Autoboot 5th selection from SETUP command 6
0 1 0 0 0 X X X	10X	Autoboot 4th selection from SETUP command 6
0 0 1 1 0 X X X	06X	Autoboot 3rd selection from SETUP command 6
0 0 1 0 0 X X X	04X	Autoboot 2nd selection from SETUP command 6
0 0 0 1 0 X X X	02X	Autoboot 1st selection from SETUP command 6
0 0 0 0 0 X X X	00X	Run standalone mode tests in a loop at power-up

The following list specifies the default values for SB 1 to SB 6 if the EEPROM is initialized in setup mode. You may change the values to any you desire by using setup mode command 6.

SB 2	DL0	RL01, RL02
SB 3	MS0	TS11, TU80
SB 4	MU0	TU81, TK50
SB 5	E	(Not set)
SB 6		(Not set)

Table C-1 shows how to select the baud rate for the console SLU when using the baud rate switch. Switches 6-8 on the KDJ11-BF module must be off to allow the baud rate switch to correctly select the baud rate.

Table C–1 Baud Rate Selection

Baud Rate Selected	Switch Position	BCR Register Bits <2:0>
38,400	0	0 0 0
19,200	1	0 0 1
9,600	2	0 1 0
4,800	3	0 1 1
2,400	4	1 0 0
1,200	5	1 0 1
600	6	1 1 0
300	7	1 1 1

Table C–2 shows how to select the baud rate with the switches on the KDJ11-BF module if the baud rate switch is not present or has to be removed due to failure.

Table C–2 KDJ11-B Switch Selection (1 = OFF, 0 = ON)

KDJ11-B Module Switch		Data Read from BCR Register
6 7 8	Baud Rate	2 1 0
0 0 0	38,400	0 0 0
0 0 1	19,200	0 0 1
0 1 0	9,600	0 1 0
0 1 1	4,800	0 1 1
1 0 0	2,400	1 0 0
1 0 1	1,200	1 0 1
1 1 0	600	1 1 0
1 1 1	300	1 1 1

Appendix D Floating-Point Instruction Timing

Since the FPJ11 is a coprocessor operating in parallel with the J-ll chip set, the calculation of floating-point instruction times for J-11 systems (using the FPJ11 option) must take this parallel processing into account.

Part	Definition
FPJ11 cycle	Two clock periods (110 ns at 18 MHz)
J-11 nonstretched cycle	Two FPJ11 cycles (220 ns at 18 MHz)
J-11 read cycle	J-11 nonstretched cycle if cache hit. Dependent on read access time of system if cache miss. The minimum is two J-11 nonstretched cycles, after which the J-11 stretches in half-cycle increments until MCONT is asserted.
J-11 write cycle	Dependent on write access time of system (two J-11 cycles + half cycles until MCONT).
Instruction Decode	A decode/prefetch cycle followed by a MOV microinstruction that allows the FPJ11 to assert DMR prior to the start of the next microinstruction (INPR for REG mode). This time equals two nonstretched cycles if the prefetch is a cache hit; otherwise nonstretched plus one read cycle.
Address Calculation Time	J-11 time required to calculate the address of the operand. This time is dependent on the addressing mode of the instruction, the frequency of the system clock, and whether any indirect data required is present in the cache (see Table D-1).
Argument Transfer Time	J-11 time required to load or store floating-point operands. This time is one nonstretched cycle (address relocation μcycle) plus one read cycle per 16-bit word read from memory for load class instructions, or one nonstretched plus one write cycle per 16-bit word to memory for store class instructions.
INPR (FEATEMP,TEMP)	J-11 support code microinstruction execute for all FPJ11 instructions. Moves the PC of the previous FPJ11 instruction to a TEMP register in case that instruction resulted in a floating-point exception. If the FPJ11 is still executing the previous instruction when the J-11 reaches its INPR microinstruction, the FPJ11 asserts STALL causing the J-11 INPR μcycle to stretch. The J-11 then waits for the FPJ11 to deassert STALL, signaling the system interface to assert MCONT before executing the next microinstruction (OUTR).

Part	Definition
WAIT	J-11 time waiting for the completion by the FPJ11 of the previous FP instruction. For load class or REG mode instructions, the time from when the J-11 INPR cycle stretches at the trailing edge of male until the FPJ11 deasserts STALL. This time equals zero if a stall was not required or if the FPJ11 deasserted the stall signal after the INPR cycle began but prior to the trailing edge of male. Although the wait time for the latter case is zero, RESYNC time is required. For store class instructions the wait time equals the time between the assertion of SCTL (that is, when the system interface is ready to execute the first write cycle of an FP store) and the assertion of FPA-RDY (data ready) by the FPJ11.
RESYNC	For load class and REG mode instructions the time required to continue a stretched INPR. This is the time for the system interface to recognize the deassertion of STALL and assert MCONT, plus the time required for the J-11 to synchronize MCONT and advance to the next microinstruction. Store class instructions normally do not have RSYNC time since the J-11 is waiting in a stretched write cycle and the continuation time is part write cycle. However, if the FPJ11 is executing a previous MODF/D or DIVD, the FPJ11 will assert STALL in order to stretch a non-I/O cycle prior to the first bus write. This allows the system interface to service DMA, thus limiting the worst case DMA latency when waiting for FPJ11 output. In this case, a wait and RESYNC time associated with the stretched non-I/O cycle is added to the effective execution time of the store class instruction.
OUTR (PC,FEATEMP), TESTPLA FPE	Last J-11 support microinstruction unless there is an FPE from the previous FP instruction. Saves address of PC in FEATEMP.
PRDC SYNC	Time required by FPJ11 to decode FP instruction and begin execution after receiving PRDC. This time equals two or three FPJ11 cycles depending upon synchronization. PRDC SYNC is not added to FPJ11 instruction execution times when the FPJ11 is executing a previous FP instruction at the assertion of PDRC.
Floating-Point Execution Time	Time required by FPJ11 to complete an FP instruction once it has received all arguments. For store class instructions, floating-point execution time includes the time from the start of the instruction until the FPJ11 asserts FPA-RDY, indicating the first 16-bit word is available for output (see Table D-2).
Effective Execution Time	Total J-11 time required to execute an FP instruction.
Load class	Instruction Decode + Address Calculation + Argument Transfer + INPR + WAIT + RSYNC + OUTR
REG mode	Instruction Decode + INPR + WAIT + RSYNC + OUTR
Store class	Instruction Decode + Address Calculation + INPR + Argument Transfer + WAIT + OUTR

Table D-1 shows address calculation times. Table D-2 shows floating-point instruction times.

Table D-1 Address Calculation Times

Mode	Load Class	Store Class
0	0	0
1	3	3
2	3	2
3	3 + RD[1]	2 + RD
4	4	4
5	3 + RD	3 + RD
6	3 + RDI[2]	2 + RDI
7	3 + RDI + RD	3 + RDI + RD
27	2	2
37	2 + RDI	1 + RDI
67	3 + RDI	2 + RDI
77	4 + RDI + RD	4 + RDI + RD

[1]RD = J-11 Read Cycle

[2]RDI = J-11 Istream Request

Table D-2 FPJ11 Instruction Times

Instruction	Min Cycles	Typ Cycles	Max Cycles	Stretch Cycles[1]	18 MHz[2] Typ(μs)
ADDF/SUBF	7	9	19	5	1.0
ADDD/SUBD	7	9	30	5	1.0
MULF	15	15	16	11	1.7
MULD	26	26	27	22	2.9
DIVF	17	24	30	25	2.7
DIVD	33	48	62	57	5.4
MODF	28	34	43	15	3.7
MODD	39	45	71	26	5.0
CMPF/D	3	4	6	2	0.4
LDF/D	3	3	3	0	0.3
LDEXP	2	3	2	0	0.2
LDCIF/D	10	10	10	3	1.1
LDCLF/D	10	10	10	3	1.1
LDCFD	4	4	4	1	0.4
LDCDF	4	4	8	1	0.4
STF/D	3	3	3	0	0.3
STCFI	8	10	13	1	1.1
STCFL	8	12	16	1	1.3
STCFD	4	4	4	0	0.4
STCDF	6	6	6	1	0.7
STEXP	5	5	5	0	0.6
TSTF/D, LDFPS STFPS, CFCC, SET	3	3	3	0	0.3
ABSF/D, NEGF/D	4	4	5	0	0.4

[1]Stretch cycles indicate the number of cycles out of max cycles that a data dependent stretch of one additional cycle could occur with probability less than 1 percent for each additional cycle.

[2]18 MHz = 111 ns Cycle

Load class instructions require input data and deposit results to the destination FP accumulator. REG mode instructions are FP accumulator to FP accumulator.

Execution of a load class FP instruction by the FPJ11 occurs in parallel with J-11 operation and can be overlapped as shown in the following flow.

J-11	FPJ11
Load class instruction is prefetched. This occurs during previous instruction execution	
Instruction Decode Prefetch next instruction	PRDC SYNC
Address Calculation	
Argument Transfer	FPJ11 loads operands
INPR	FPJ11 execution starts
WAIT if any	
RSYNC if any	
OUTR	
Decode next instruction	
	FPJ11 only stalls if next instruction is FP and REG mode. The FPJ11 can overlap the loading of operands for subsequent load class instructions.
	FPJ11 execution unit done

Store class instructions can be overlapped by the J-11 as the FPJ11 will complete a previously started load class or REG mode instruction and then continue to the store instruction. Execution of the store class instruction must be completed before the result can be stored in memory, thus eliminating further parallel processing for store class FP instructions. See the following flow.

J-11	FPJ11
Store class instruction is prefetched. This occurs during previous instruction execution	
Instruction Decode Prefetch next instruction	PRDC SYNC
Address Calculation	FPJ11 starts execution
INPR Argument Transfer	FPJ11 places operands in output buffer and sets FPA_ RDY
J-11 waits during first write if FPA-RDY not asserted	
J-11 completes argument transfer	
OUTR	
Decode next instruction	

Appendix E
ROM Code Differences

E.1 General

The KDJ11-BF module uses two ROMs that contain the boot and diagnostic coding described in Chapter 3. The original version is designated as V6.0 and the revised or updated versions are V7.0 and V8.0. The user does not have to remove the module from the system for identification because the version number is shown in the upper right hand corner of the display whenever setup mode is entered. The ROM part numbers associated with each version are shown in Table E–1. The differences between V6.0 and V7.0 are detailed in Section E.2, while the differences between V7.0 and V8.0 are covered in Section E.3.

Table E–1 ROM Part Numbers

Socket	V8.0 Set	V7.0 Set	V6.0 Set
Low byte E116	23-168E5	23-116E5-00	23-077E5-00
High byte E117	23-169E5	23-117E5-00	23-078E5-00

E.2 V6.0 and V7.0 Differences

E.2.1 Boot Support for Tape MSCP Devices (TK50/TU81)

V7.0 has a built-in tape MSCP boot program for the TK50/TU81 devices and the device name is MU. The tape MSCP boot and the disk MSCP boot are combined into one common boot program.

V6.0 does not have a tape MSCP boot program for the TK50/TU81 devices. UNIBUS systems could boot these devices if an M9312 type boot ROM for tape MSCP devices could be installed in the UBA module, but this type of boot ROM is not available.

E.2.2 Disk MSCP Automatic Boot Routine

In the V7.0 MSCP automatic boot, the program tries to boot removable media units from 0 to 255 and then to boot fixed media units from 0 to 255. The program attempts to boot each unit at the standard MSCP address and if this fails, the boot program attempts the same unit number from the first floating disk MSCP device (if it is present) before continuing to the next unit number. The routine always makes the first pass trying to boot the removable media units and the final pass trying the fixed media units.

In the V6.0 MSCP automatic boot (device name A), the program tries to boot removable media units from 0 to 7 and then to boot fixed media units from 0 to 7. It only tries to boot the drives attached to the controller at the standard address of 172 150. The MSCP automatic boot does not support unit numbers above 7 and it hangs if the controller has a response from a unit number greater than 7.

The first floating controller (when present) is at address 160 334, if there are no devices from 160 010 to 160 330. The main advantage of V7.0 is to allow the user to add a second disk MSCP device without making any entries into the translation table (as long as the controller address is set exactly according to the floating CSR address rules).

E.2.3 Dialog Mode Boot Command for Disk MSCP Boot

V7.0 of the dialog mode lets the user execute the boot command for a DU device and the ROM code tries to boot the selected unit number at the standard controller address. If the boot is not successful, the ROM code then tries to boot the same unit number at the first floating controller address (if it is present). When an error occurs on both controllers, the V7.0 ROM code prints out error messages for both controllers starting with the standard address. Nonexistent error messages are not printed unless the unit is nonexistent on both controllers. If the second controller does not exist at the proper floating address, the ROM code prints out messages associated with the standard controller only. When the translation table or the /A switch is used, only one controller is tried regardless of the existence of two or more controllers.

V6.0 of the boot routine tries the standard address only, unless otherwise directed by the translation table or the /A switch.

E.2.4 Disk MSCP Boot (DU)

The V7.0 disk MSCP boot always initializes the disk controllers when they are first accessed. The controller is left on-line, unless it is necessary to take it off-line. This allows the boot to operate faster in the automatic boot mode when many unit numbers and possibly multiple controllers are being tried. The controller is always turned off before control is transferred to the secondary boot. The V7.0 DU/MU boot requires a 16-Kword memory (minimum) and the V6.0 DU boot requires an 8-Kword memory (minimum).

In V6.0, the controller is initialized only when the SA register is not zero. The controllers are usually left on and are turned off before transferring control to the secondary boot. The controller is also turned off before checking for a valid boot block. Therefore, if the automatic boot sequence "sees" a lot of nonbootable media before it gets to the device being booted, the boot code may be slow since it has to reinitialize the controller after each nonbootable unit is found.

E.2.5 8-Unit Restriction for MSCP Automatic Boot

V6.0 is restricted to units 0 through 7 and if the first unit on the controller is unit 8 or greater, the boot loops because the automatic boot program does not correctly handle unit numbers greater than 7. V7.0 can handle unit numbers from 0 through 255.

E.2.6 Irregular Monitoring of Keyboard During Automatic Boot Sequence

As the ROM code proceeds through the devices during the V6.0 automatic boot, it does not check the keyboard for a Ctrl C unless a specific boot program does it. The keyboard is sometimes checked by a boot when the boot program is in a potentially long loop waiting for some action to occur. V7.0 checks the keyboard at least once between each boot in the automatic boot sequence.

E.2.7 Addition of Single-Letter Mnemonic in Automatic Boot List

A single-letter mnemonic (L) has been added to the boot command list for V7.0. The L command causes the automatic boot sequence to loop continuously until one of the selected devices is successfully booted. Normally, the last device in the automatic boot table is followed with the mnemonic E, which causes the sequence to exit at the end of the table, and if no device is successfully booted, the ROM code displays an error message and requests input before proceeding.

When the L follows the last device, the ROM code restarts the table at the beginning and continuously tries each device in the table until one is booted or the user types Ctrl C to abort the sequence. This feature is useful for booting a fault-tolerant system that must be tried continuously until a successful boot occurs.

The L command is not included in V6.0, but the user can implement it by writing a small EEPROM boot to emulate the feature. The source code and the description of this program (to enable a continuous loop function for V6.0) are shown in Example E-1. When this feature is implemented, it must be noted that there is no boot program using a device name of L, and if there is, the user has to delete or rename that boot before using the new program.

E.2.8 Setup Mode Disable

V7.0 includes a disable parameter on the list of parameters used by setup command 2. This command was added to prevent unauthorized entry into setup mode and it allows the user to disable entry into setup mode if the forced dialog mode is not selected. This change assumes that the forced dialog mode switch is controlled or that switch 5 on the module is on to prevent unauthorized entry into setup mode. When the ROM code is in dialog mode and setup mode is disabled, all references to the setup commands are eliminated, and typing SETUP causes an invalid command response from the ROM code. In V6.0, the setup mode can always be entered from dialog mode.

```
.=10000                                 ; Program is relocatable to another
                                        ; address.

START:    tstb    @#177560              ; Has any characters been typed
          bpl     10$                   ; No-Go exit back to auto boot
                                        ; Yes-Check the character
          movb    @#177562,r5           ; Get the character from the RBUF
          bic     #177600,r5            ; Clear off all bits above bit 07
          cmp     r5,#3                 ; Is the character a CTRL C?
          beq     20$                   ; Yes-Then return to ROM code with
                                        ; r5 set to 3 which will cause the
                                        ; boot sequence to be aborted.

10$:      mov     #301,r5               ; Load r5 with value for drive error
          movb    #100,@#177611         ; This will fake out the ROM code and
                                        ; make it restart the auto boot sequence

20$       bic     #760,@#177520         ; Make sure the ROMs are selected in
                                        ; the BCSR
          jmp     @#165762              ; Return to the ROM code.
                                        ; If r5 is 301 then restart the auto boot
                                        ; sequence. If r5 is 3 then abort the
                                        ; sequence and go to Dialog mode.
```

Example E-1 Program for Continuous Loop

E.2.9 Disable All Testing Parameter

V7.0 includes a disable testing parameter on the list of parameters used by setup command 2. When this parameter is set or selected, it disables all memory and cache testing if the forced dialog mode is not selected. (The forced dialog mode causes the module to run the complete set of tests.) This reduces the testing time to approximately 70 or 85 ms. This parameter is not available in V6.0.

E.2.10 Edit/Create Command

In V7.0, the edit/create command of the setup mode uses a decimal value for the highest unit number entry on the EEPROM boots. V6.0 uses an octal number that is converted into a decimal number.

E.2.11 Initialize Command for the PMG Counter

The initialize command sets the PMG count value to 7 in V7.0. This value was set to 0 in V6.0. The recommended value for the PMG count is 7 for all modules that use V6.0.

NOTE
It is recommended that users of V6.0 change the PMG count value from the default value of 0 to a value of 7.

E.2.12 PMG Parameter Warning

V7.0 prints a warning message if the PMG count value is set to 0 by the user. The warning was created to prevent the user from operating the system with a PMG count value of 0. This ensures that the CPU is not locked out from the bus for excessively long periods of time, which could cause some loss of data if it is stalled for more than 250 ms. The message shows the PMG count value being changed and prints the warning with the parameter line being reprinted, allowing the user to change the PMG count value. The display also contains the current values associated with the selections available to the user and thus eliminates the need to consult a reference document. V6.0 prints only the parameter selected and the values the user may select.

V6.0 PMG count parameter printout

```
PMG count                              (0-7) = 7 NEW =
```

V7.0 PMG count parameter printout

```
PMG 0--(7) 1 = 4us, 2=8, 3 = 1.6, 4 = 3.2, 7 = 25.6 = 7 NEW =
```

E.2.13 Setup Command 4 Printout

V7.0 prints descriptions of the single-letter mnemonics A, B, E, and L when they are used by setup command 4. V6.0 prints only the descriptions for A and E because there are no descriptions for B and L. The V7.0 descriptions are shown in Example E–2.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 4 Return

List/change the Automatic boot selections in the Setup table

A = MSCP automatic boot
B = External ROM boot
E = Exit automatic boot
L = Loop continuously

Boot 1 = A
Boot 2 = DL0
Boot 3 = MS0
Boot 4 = MU0
Boot 5 = E
Boot 6 = blank

Type Ctrl Z to exit or press the RETURN key for No change

Boot 1 = A
Device name =
```

Example E–2 Single-Letter Description for Command 4

E.2.14 MU (TK50/TU81) Device

V7.0 adds the device name MU for the TK50 or TU81 to the list of devices in the automatic boot selections table. This is also added to the list when the setup mode initialize command is executed. V6.0 does not have the MU device name. The setup command 4 automatic boot lists are shown in Table E–2 for both versions.

Table E-2 Setup Command 4 Automatic Boot Lists

V7.0	V6.0
A	A
DL0	DL0
MS0	MS0
MU0	A
A	

E.2.15 Setup Command 5

Setup command 5 is eliminated in V7.0. The setup command 5 description is reserved and if the command is selected, it is ignored.

In V6.0, this command allows different character sets in the console terminal to automatically be selected by the ROM code when the user changes from English to a local text or from local to English text. The command is no longer required since all text printed on the screen uses only the standard ASCII characters generally available on all terminals. Special characters used in some languages are imitated by fallback representations in standard ASCII.

E.2.16 Memory Initialization at Power-Up

V7.0 writes to all consecutive memory starting at location 0 at least once after the power-up sequence is complete. This feature is disabled if the disable testing option is enabled. This option does not apply to restarts. V6.0 may not write to locations above 248 Kbytes if the long memory test is disabled or Ctrl C is typed.

E.2.17 Power-Up Option Set to 3 with Battery Backed Up Memory

In V7.0, if the selected power-up mode is 3 and the battery indicates that the voltages are lost with the ignore battery function turned off, the ROM code goes to the dialog mode regardless of the restart mode selection. For the same conditions in V6.0, the ROM code executes the restart mode selection if it is not mode 3 or it goes to the dialog mode. The battery OK signal is currently used only in UNIBUS systems.

E.2.18 Halt-on-Break

V7.0 sets the halt-on-break bit in the BCSR immediately after the "Testing in progress—Please wait" message is displayed. The halt-on-break feature, generally used in single-user environments, was not needed.

V6.0 does not set the halt-on-break bit in the BCSR until a break is received and discarded, any valid character is received except XON, or the ROM code gives up control of the CPU. This allows the ROM code to ignore any breaks that come as a result of a terminal being powered up.

E.2.19 Local Language Support

V7.0 supports local language translations by using the CTRL L command. Local language is not supported in V6.0.

E.2.20 Addition of Map Command Feature

V7.0 adds an additional feature to the map command. This feature determines the clock speed of the CPU by counting the number of SOB instructions that can be executed out of the cache memory during one 20 ms cycle of the internal DLART clock. This value is compared with a table of standard values and if it is within 0.1% of any standard value, that value is displayed. If it does not match a standard value, the actual value is displayed. The standard values are 15.206, 17, 18, 19, and 20. The speed is not calculated if any errors are detected during testing.

E.2.21 EEPROM Load Error Before Dialog Mode

In V7.0, if the setup mode is entered and an error occurs in loading the EEPROM data into memory, the dialog mode is restarted and no error message is generated. V6.0 does not check to verify the data is OK and setup mode cannot be entered without testing memory. In either case, a timeout may occur and trap to location 4 with an error message being generated.

E.2.22 Test Command Decimal Numbers

In V6.0 dialog mode, when the user selects a specific test with the test command, the ROM code selects a different test number. The valid test numbers are in the range of 31 to 70 (octal) with the exception of tests 64 to 66 and any UNIBUS test on LSI-11 bus systems. The only test numbers that may cause confusion are illegal test numbers that end in 8 or 9 using the decimal system. Table E-3 lists the selectable (illegal) test numbers and the actual test run by the ROM code.

Table E-3 ROM Code Test Selections

Selected Test	Actual Test
78	70
69	61
68	60
59	51
58	50
49	41 (UNIBUS only)
48	40 (UNIBUS only)
39	31 (UNIBUS only)

V6.0 *does* echo the correct and actual test being run. For example, if the user selects T 59, then V6.0 responds that it is looping on test 51. V7.0 corrects the problem.

E.2.23 Test Command Execution of a Single Test

In V7.0, if the test command is used and a specific test is selected, the memory size routine is run before the selected test is run. Some of the memory size parameters have been lost and need to be replaced. V6.0 runs only the selected test when a single test is selected.

E.2.24 Test Errors in Tests 72 to 75

In V6.0, if an error occurs in tests 72 to 75, the user has a choice of either rerunning the test or looping on the test. It does not matter what the user selects, however, because the ROM code unconditionally restarts from the beginning if the user selects a valid choice. For V7.0, the user is only allowed to rerun the test, but the ROM code still restarts the code from the beginning when this selection is made.

E.2.25 Bypass Errors for Test Failures

In V6.0, if an error occurs during testing, the user may bypass the test if the error is considered to be nonfatal. Many times it is difficult to determine if an error is fatal or nonfatal and sometimes, if an error is determined to be nonfatal, it may still cause a problem when overridden.

V7.0 considers all errors to be fatal and never provides the override command. However, the user can still override the error in the same way used for V6.0. To override, the user types Ctrl O and then types 4 Return. If Ctrl O is not typed, the 4 is ignored and not echoed.

E.2.26 Test 76 and 75 Error Messages

During the first two major tests, 76 and 75, the printout for errors has been changed. These tests have a simple printout because the normal printout routine has not been turned on at this time. V6.0 prints "Error 76" or "Error 75" and V7.0 prints "A 76" or "A 75." This change was made for local language applications since these printouts cannot be translated.

E.2.27 Starting Automatic Boot Sequence Message

V7.0 prints a message indicating when the automatic boot mode is selected and the sequence is starting. This message (Example E-3) indicates that all the testing is complete and the ROM code is starting the automatic boot sequence.

NOTE
This does not apply to LSI-11 systems with the friendly format feature selected by setup command 2.

```
Testing in progress - Please wait
Memory size is 512 K Bytes
9 Step memory test
Step 1 2 3 4 5 6 7 8 9

Starting system
```

Example E-3 Automatic Boot Sequence Message

E.2.28 Device Name and Number After Message

V7.0 prints the device mnemonic and unit number after the "Starting system" message shown in Example E-4. This has no affect on the printout when the user friendly printout feature is selected.

```
Testing in progress - Please wait
Memory size is 512 K Bytes
9 Step memory test
Step 1 2 3 4 5 6 7 8 9

Starting automatic boot

Starting system from DUO
```

Example E-4 V6.0 Incorrect Message

E.2.29 Incorrect Message for Invalid Unit Number

V6.0 responds with an incorrect message (Example E-4) when the user types in a unit number greater than 255. V7.0 corrects this problem by printing a message (Example E-5) indicating the invalid unit number.

```
Commands are Help, Boot, List, Setup, Map, Test.
Type a command then press the RETURN key: B DL300 Return

Invalid unit number

Commands are Help, Boot, List, Setup, Map, Test.
Type a command then press the RETURN key:
```

Example E-5 V7.0 Correct Error Message

E.2.30 Dialog Mode Header Message

V7.0 changes the dialog mode header message by deleting the brackets because they are not available on all terminals.

E.2.31 Map Command Message

V7.0 changes the "&" symbol to "and" for the map command message in setup mode because the symbol is not available on all terminals.

E.2.32 List Device Descriptions

V7.0 changes the descriptions for the device names in some of the mnemonics and also shows the TK25 and TS05 devices under the mnemonic MS for UNIBUS systems. The differences are not listed here because they are obvious. RA80/81/60, for example, is changed to RA80, RA81, and RA60.

E.2.33 Loss of the First Line in a List Header

In V6.0 dialog mode, when the user types the boot command without the device and then types Return ? to get a list of boot devices, the ROM code does not send a line feed before the header of the list and the header is lost in the right margin (Example E-6). The list is typed out correctly. V7.0 corrects the problem and the message shown in Example E-7 is displayed.

```
Commands are Help, Boot, List, Setup, Map, Test.
Type a command then press the RETURN key: B Return

Enter the device name and unit number then press the RETURN key: ?

name        numbers       Source       Device type

DU          0-255         CPU ROM      MSCP RA80/81/60, RD51/52, RX50, RC25...
DL          0-3           CPU ROM      RL01/RL02
            etc....
```

Example E-6 V6.0 List Header Error

```
Commands are Help, Boot, List, Setup, Map, Test.
Type a command then press the RETURN key: B Return

Enter the device name and unit number then press the RETURN key: ?

Device      Unit          CPU ROM      (RA80, RA81, RA60, RD51, RD52, RX50, RC25)
name        numbers       CPU ROM      RL01/RL02

DU          0-255 DL
DL          0-3
            etc....
```

Example E-7 V7.0 Correct List Header

E.2.34 CTRL R and CTRL U Echo

V6.0 echoes the CTRL R and CTRL U commands as R and U, respectively. V7.0 does not echo these commands because the symbols are not available on all terminals. These commands still function the same way.

E.2.35 Power-Up or Restart Mode Set to 3 (LSI Bus Only)

In V6.0, before executing a power recovery trap through location 24, the ROM code does the following.

1. Reads and stores the contents of location 24
2. Executes a read/write test on location 24
3. Restores the original contents of location 24

When the test is successfully completed, the ROM code loads the contents of location 26 into the PSW and jumps to the location specified in location 24. Since this location was tested, the ROM code cannot be present in the lower portion of memory.

V7.0 does not test location 24 and it is possible to have ROM code in the lower portion of memory. The ROM code loads the contents of location 26 into the PSW and jumps to the location specified in location 24.

E.2.36 Automatic Boot Misleading Error Message (LSI Bus Only)

In V6.0, R5 is cleared at the end of the MSCP disk sniffer boot and this causes all errors that occurred during the sniffer to appear to be correctable by the user. This minor problem only affects the message sent to users operating in the friendly mode. V7.0 corrects the problem.

E.2.37 APT Halt-on-Break Detect (LSI Bus Only)

V6.0 can detect breaks coming from an APT system. This feature allows LSI type systems that have the halt-on-break option disabled to halt, and enables the halt-on-break option if the APT is trying to down-line-load. V7.0 eliminates this feature because it is implemented in the manufacturing process. If the feature is not eliminated, there is a small chance that the system may be halted with halt-on-break disabled if the terminal is a VT5X terminal (or possibly other terminals), but not if it is a VT1XX or VT2XX terminal.

NOTE

Note also that autobaud detect routines from remote hosts can cause halt-on-break when it is not desired.

E.2.38 B Mnemonic for ROM Boots (UNIBUS Only)

For V7.0 under the B mnemonic for ROM boots, the address located at 173 024 on the M9312 module in the UNIBUS system must be an even address. This is the only check of the address data. For V6.0 under the B mnemonic for ROM boots, the address located at 173 024 must be 165 000 or greater, but it can be odd. In either case, if all the conditions are not met, an invalid device message is reported.

E.2.39 Error in List Command When Unknown ROM is Found (UNIBUS Only)

In V6.0, the ROM board for the UNIBUS must respond to all addresses from 17 773 000 to 17 773 776 for the ROM code to transfer control using the B mnemonic, or else an invalid device message is reported. In V7.0, only address 17 773 024 must respond.

E.2.40 Power-Up or Restart Mode Set to 3 (UNIBUS Only)

In V6.0, the ROM code checks for the presence of UNIBUS memory and sets up the KMCR before executing a power recovery trap through location 24. Then the ROM code does the following.

1. Reads and stores the contents of location 24
2. Executes a read/write test on location 24
3. Restores the original contents of location 24

When the test is successfully completed, the ROM code loads the contents of location 26 into the PSW and jumps to the location specified in location 24. Since this location was tested, the ROM code cannot be present in the lower portion of memory.

V7.0 does not check for UNIBUS memory and assumes that by selecting mode 24 the system has the final configuration of memory already installed. Therefore, location 24 is not tested and it is possible to have ROM code in the lower portion of memory. The ROM code loads the contents of location 26 into the PSW and jumps to the location specified in location 24.

E.2.41 Saving KMCR Bits <5:0> in the EEPROM (UNIBUS Only)

In V7.0, when the setup table is written into the EEPROM, the contents of KMCR bits <5:0> are always copied into the EEPROM regardless of the power-up or restart modes. The EEPROM data is used to load the KMCR when the ODT or 24/26 modes are selected. The ROM code autosizes for UNIBUS memory when the automatic boot or dialog modes are selected.

In V6.0, KMCR bits <5:0> are copied into the EEPROM only when ODT is selected for the power-up or restart mode. The ODT mode is the only mode that does not autosize for UNIBUS memory and consequently must depend on the EEPROM to contain the correct KMCR information.

E.3 V7.0 and V8.0 Differences

This section describes the changes made when V8.0 of the ROM code was created. The changes made in V7.0 (as described in Section E.2) are still true for V8.0 except as noted below. Section E.2 describes the differences between V6.0 and V7.0 only.

E.3.1 M9312 MultiROM Bootstrap Support (PDP-11/84 Only)

V6.0 and V7.0 do not support M9312 bootstrap programs, which require more than one ROM to implement (multiROM bootstraps). The only way these programs can be supported for V6.0 and V7.0 is to use a work-around program loaded into the EEPROM. (See Appendix F.) V8.0 corrects this problem and automatically supports M9312 multiROM bootstraps.

NOTE
This problem occurs only in PDP-11/84 systems.

E.3.2 RAnn Disk Spinup Time Delay for Automatic Boot

In V6.0 and V7.0, the disk MSCP bootstrap assumes that off-line error codes from the disk being booted are correct. If the disk is an RAnn on a UDA/KDA controller, and if the disk is spinning up or down, it may incorrectly identify a disk spinning up as being off-line (not available). This causes the ROM code to skip this unit and try another even though there is no problem.

V8.0 works around this problem with the following strategy. It identifies the controller as a KDA50, UDA50, or UDA50A. The identification is done in step 4 of the initialize sequence. If the device is not a KDA/UDA controller, the delay is not present. If the controller type is UDA/KDA and the response packet from the controller is an off-line code (3), the ROM code repeatedly tries to boot the device for a period of at least 60 seconds. RAnn devices need this delay time to spin up and be ready to respond to the host. If the RAnn is not ready to be booted after 60 seconds, the code reports the error and sets a flag preventing this delay time from occurring again unless the code is rebooted. The next device specified in the automatic boot setup table is then tried. The code responds to the terminal shortly if it cannot find a bootable device.

In a case where the code enters dialog mode, it is assumed that the user has the RAnn spun up and ready. The code does not wait 60 seconds for the device to spin up. The device promptly reports any errors if they occur. Some RAnn devices (possibly RA60) work adequately without this change.

CAUTION

When a system is configured with RAnn disks (and possibly with other non-RAnn MSCP disks), it is important to realize that the wait loop in V8.0 delays the automatic boot process for 60 seconds or more. It is recommended that the user remove A (disk MSCP automatic boot) from the boot table in the EEPROM using setup command 4. The user should replace it with the desired order of devices to be booted (such as DU0, DU2, and so on). This is especially true when booting fixed media devices, since the disk automatic boot ignores fixed media devices until it has tried all removable media devices.

Remember that the disk automatic boot tries each unit at the standard controller address and then at the first floating address. This is also true for individual unit numbers (such as DU2, DU0, and so on) unless the unit number is described in the translation table (setup mode command 3).

E.3.3 Addition of RESET Instruction at Beginning of Code

V8.0 executes a RESET instruction (bus reset) at the beginning of the code. V6.0 and V7.0 do not include this instruction. This change provides a bus reset after POK is asserted.

E.3.4 Addition of New Setup Command 5

V8.0 adds a new setup mode command 5, similar to the setup mode command 5 in V6.0. V7.0 does not have a setup mode command 5. This new command in V8.0 allows the user to store up to 20 bytes of information in the EEPROM. The data is stored in the same location in the EEPROM as for V6.0. However, the information stored there is never sent to the console as it was in V6.0. The data must be entered as octal numbers in the range of 0 to 377. This command may be used to store information such as serial numbers. The setup mode initialize command resets this data to 0. The ROM code does not use this data for any purpose at all.

E.3.5 Memory Test Coverage Problem

V6.0 and V7.0 test only the first 4 Kwords of memory when running test 50. V8.0 corrects this and checks all available consecutive memory. Test 50 checks two locations for floating 1s and 0s and does byte testing.

E.3.6 List Command Device Descriptions

Some of the messages printed during the V8.0 list command are new. The changes are given in Table E-4.

Table E-4 New List Command Device Descriptions

Message Type	From:	To:	Comments
DU	RD51, RD52, RC25, RA80, RA81, RA60	RDnn, RXnn RC25, RAnn	
XH	DECnet DEQNA	DECnet Ethernet	11/73 or 11/83 only
XE	DECnet DEQNA	DECnet Ethernet	11/84 only (if ROM present)

E.3.7 Manufacturing Test Loop Problem

The manufacturing test loop in V7.0 does not execute all of its tests. V8.0 corrects the problem. V6.0 always worked correctly. This change affects only those manufacturing sites that use the feature. The manufacturing test loop can only be selected by using the switchpack on the CPU module.

Appendix F
Multiboot ROM Control Transfer

F.1 Introduction

In V6.0 and V7.0 of the ROM code, the routine that identifies ROMs on the UBA or the M9312 will not correctly identify boot ROMs that use more than one ROM for the boot. Because of this, these boots will not list and cannot be started from the base ROM code without using a small EEPROM boot program to transfer control to the ROMs. This problem will be corrected in any future releases of the CPU ROM code.

F.2 Transfer Control Program

The following is the simplified program used to transfer control to any UBA ROM or M9312 ROM boot if needed. The starting address in location 010020 may have to be changed depending on the ROM and the socket it is located in.

```
bcsr   = 177520
dcsr   = 177730

010000      052737    bis   #200,@#bcsr          ; disable CPU ROMs in
010002      000200                               ; 173nnn address range
010004      177520                               ;
010006      042737    bic   #10,@#dcsr           ; make sure UBA ROMs
010010      000010                               ; are enabled
010012      177730                               ;
010014      000261    sec                        ; disable diagnostics
010016      000137    jmp   @#173012             ; go start boot for ROMs
010020      173012                               ; in sockets 1-3 of UBA.
                                                 ; Change 16/173012 to
                                                 ; 173212 if ROMs are in
                                                 ; UBA sockets 2-4
```

NOTE
If ROM is on M9312 module, then change 010006 from 042737 to 052737.

NOTE

If ROM is not in socket 1, then address in 10020 must be adjusted by 200 for each socket as follows:

address = 1730nn for socket 1
address = 1732nn for socket 2
address = 1734nn for socket 3
address = 1736nn for socket 4

The previous program works for the DECnet ROM boots for DMC11/DMR11, DUP11, DU11 and DL11-E with the following part numbers. The user should type in the correct device name and device description, as follows, when the program is being loaded into the EEPROM under setup mode of the CPU ROM code.

ROM Part Numbers	Device Name	Device Description
23-862A9, 23-863A9, 23-864A9	XM	DMC11/DMR11
23-865A9, 23-866A9, 23-867A9	XW	DUP11
23-868A9, 23-869A9, 23-870A9	XU	DU11
23-926A9, 23-927A9, 23-928A9	XL	DL11-E

The same general program could be used for any multi-ROM boot or any single ROM boot that does not follow the M9312 ROM format standards. The starting address may have to be adjusted to start the boot.

F.3 EEPROM Load Example

Example F-1 shows a sample program being loaded into the EEPROM for the DECnet DMC11/DMR11 boot ROMs.

```
KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 13 Return

Edit/create an EEPROM boot

Type Ctrl Z to exit or press the RETURN key for No change

1410 Bytes free in the EEPROM

Device Name              = AA          New = XM
Beginning address        = 000600      New = 10000
Last byte address        = 000615      New = 10021
Start address            = 000600      New = 10000
Highest Unit number      = 3           New = 15
Device description       =             New = DMR11/DMC11
```

Example F-1 EEPROM Load

```
ROM ODT> 010000/000000  052737
ROM ODT> 010002/000000  000200
ROM ODT> 010004/000000  177520
ROM ODT> 010006/000000  042737
ROM ODT> 010010/000000  000010
ROM ODT> 010012/000000  177730
ROM ODT> 010014/000000  000261
ROM ODT> 010016/000000  000137
ROM ODT> 010020/000000  173012
ROM ODT> 010022/000000  ^Z

KDJ11-B Setup mode
Press the RETURN key for Help
Type a command then press the RETURN key: 14 [Return]

Save boot into the EEPROM

Are you sure? 0=No,   1=Yes
Type a command then press the RETURN key: 1 [Return]

Writing the EEPROM - Please wait
```

Example F-1 (Cont.)   EEPROM Load

Index

Digital Equipment Corporation • Maynard, MA 01754